TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

ARVIN JALALI

INTERACTIVE PLANNING TOOL FOR GLOBAL SOFTWARE PROJECTS

Master of Science Thesis

Examiners: Prof. Kari Systä, Prof. Outi Sievi-Korte
Examiner and topic approved by the Faculty Council of the Faculty of Computing and Electrical Engineering on 6th May 2015.

ABSTRACT

Tampere University of Technology
Master's Degree Programme in Information Technology
ARVIN JALALI: Interactive Planning Tool for Global Software Projects
Master of Science Thesis, 60 pages
April 2016
Major: Software Systems
Examiners: Professor Kari Systä, Professor Outi Sievi-Korte
Keywords: Global Software Development, Software Planning, Search-Based Software Engineering, Search-Based Global Software Planning Tool

When planning global software development (GSD) projects, project managers often face decision-making problems such as how to choose the most suitable teams among the available teams, or how to assign work optimally to the selected teams considering both the duration and cost of the project. There are many alternative solutions, and each of them affects the duration and cost of the project differently. Hence, the idea of developing an automated planning tool that guides the project manager is helpful. This thesis documents a research focusing on planning GSD projects by an automated tool proposed by Sri Vathsavayi at Tampere University of Technology.

Vathsavayi in his research proposes a GSD model and some preliminary ideas for developing an automated planning tool. He utilizes multi-objective genetic algorithms to apply search-based software engineering in planning GSD projects. In the thesis, the GSD model and the planning tool proposed by Vathsavayi are evaluated in few specific aspects. Firstly, the GSD model besides the planning tool is referenced to some of the typical issues of a GSD project. Moreover, some recommendations for the assessment of communication distances between teams participating in a GSD project are proposed.

The thesis, as a constructive research, focuses mostly on designing and implementing a concrete user interface (UI) for the proposed planning tool. The developed UI is a simple, clear, and user-friendly web application used by the project manager to gather data about GSD projects and participating teams. The data is stored into a database and considered as the input data to the proposed tool by Vathsavayi. Finally, the developed UI is evaluated, and some further work is proposed to provide a better user experience for the project manager.

PREFACE

This Master of Science thesis was done in the Department of Pervasive Computing at Tampere University of Technology, Tampere, Finland. The thesis presents the Interactive Planning Tool for Global Software Projects.

First of all, I would like to thank Professor Kari Systä and Professor Outi Sievi-Korte for their excellent guidance during the supervisory of the thesis.

I am also grateful to Farhoud Goudarzi, my friend in Oslo, who makes me believe in true friendship and brotherhood.

And most importantly, words cannot express my gratitude to my dear mother, Khadijeh Pashmforoosh, for her valuable support in many different aspects during my study in Finland.

Tampere, April 2016

Arvin Jalali

CONTENTS

LIST OF ABBREVIATIONS

ACO	Ant Colony Optimization
API	Application Programming Interface
CDN	Content Delivery Network
CSS	Cascading Style Sheets
GA	Genetic Algorithms
GSD	Global Software Development
HTML	HyperText Markup Language
ID	Identification
JSON	JavaScript Object Notation
PMBOK	A Guide to the Project Management Body of Knowledge
PMIS	Project Management Information System
SB-GSD	Search-Based Planning of Global Software Development
SBSE	Search-Based Software Engineering
UI	User Interface
UI-GSD	User Interface for Planning Global Software Development
URL	Uniform Resource Locator
XML	Extensible Markup Language
XSL	Extensible Stylesheet Language

1. INTRODUCTION

When planning a global software development (GSD) project, the project manager faces many decision-making situations. He/she needs to make decisions on several problems such as which teams are the most appropriate ones to be involved in the GSD project, or how to assign work optimally to the involved teams in a tradeoff between the duration and cost of the project. There are many alternative solutions, and each of them can influence the duration and cost of the GSD project differently. Hence, finding an optimal solution by the project manager without any automated support is a very difficult task. [1]

Vathsavayi in his research [1] presents a GSD model as well as some preliminary ideas for developing an automated tool support that aims to assist the project manager in planning a GSD project. In general, the proposed tool by Vathsavayi assists the project manager in planning the GSD project by exploring the optimal work allocations and the estimated cost and duration of each of them. Vathsavayi utilizes multi-objective genetic algorithms [2] to apply search-based software engineering in order to find the optimal work allocations in a GSD project. The thesis aims to design and implement a concrete user interface (UI) for the proposed tool by Vathsavayi.

1.1 Research Goals of the Thesis

The research goals of the thesis are listed in the following.

1. Literature study about GSD and its typical challenges, concept of distance in the context of GSD, GSD planning, search-based software engineering (SBSE), and the application of SBSE in planning GSD projects
2. Conducting a constructive research including iterative prototyping of the proposed tool by Vathsavayi based on regular feedback from the research team
3. Evaluation of the developed prototype and proposing some further work

1.2 Motivation

The constructive research part of the thesis focuses on designing and implementing a proper UI for the proposed tool by Vathsavayi. The developed UI aims to be used by the project manager when planning a GSD project. The proposed tool by Vathsavayi had a preliminary UI that had been made mostly for researchers to understand in general the basic idea behind the proposed tool. Hence, there was a need for developing a new UI that can be used by the project manager when planning a GSD project.

1.3 Achievements

A web application has been developed as a result of four phases of implementation. The developed application is a user-friendly and simple UI used by the project manager to gather data about the GSD project and the participating teams. The data is stored into a database. The stored data can be retrieved, updated, or deleted by the project manager at any time. Also, the data is assumed to be used by the tool proposed by Vathsavayi to plan the GSD project automatically. Figure 1 represents major parts of the developed UI.

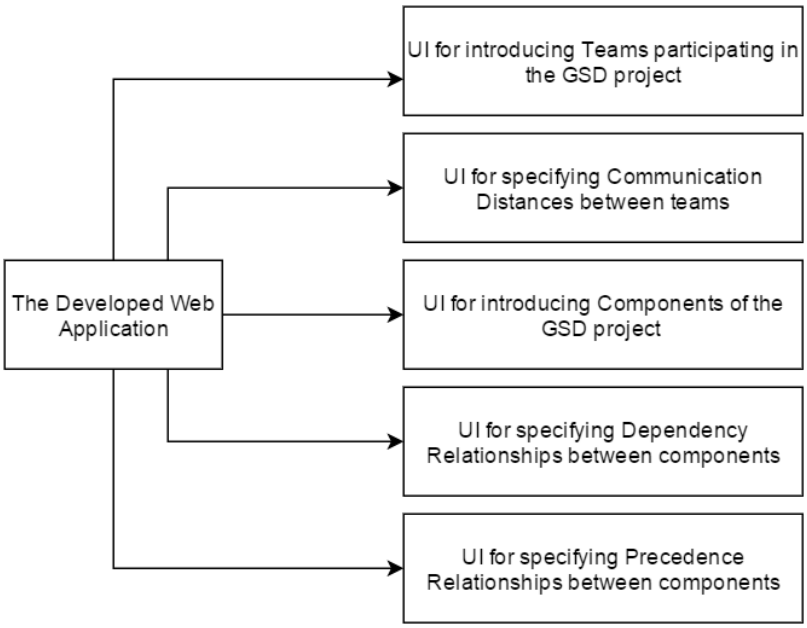


Figure 1. Major parts of the developed UI

1.4 Structure of the Thesis

The thesis consists of four major parts that are explained in the following.

1. **Literature study:** This part of the thesis was started by studying several articles about GSD and its typical challenges. Then, special focus was given to the research of Vathsavayi [1] proposing an automated tool support that assists project managers in planning GSD projects. At the same time, some literature study was conducted about some of the general background topics such as software planning and search-based software engineering (SBSE). Consequently, the more specific topics such as the concept of distance in the context of a GSD project, challenges of planning a GSD project, and the application of SBSE in planning GSD projects were studied. After completing the literature study, the proposal of Vathsavayi was evaluated in few aspects. Firstly, it was discussed how his proposal addresses some of the typical issues of GSD projects. Secondly, some recommendations for estimating the communication distances between GSD teams

were proposed. Next, the decision of prototyping the tool proposed by Vathsavayi was made in order to design and implement a concrete UI for the tool.

2. **Preparation for development of the UI:** HTML (Hyper Text Markup Language) [3], CSS (Cascading Style Sheets) [4], and JavaScript [5], as the programming languages, alongside a Firebase database [6] were chosen as the technical requirements of the development process. A certain time was spent for preparation and learning before starting the actual implementation of the web application.
3. **Constructive research:** A concrete UI was designed and implemented for the tool proposed by Vathsavayi. The development of the UI was done in four phases in an iterative and incremental model. After each iteration, a meeting was held with the research group to get feedback from them.
4. **Evaluation of the developed UI and proposing some further work:** After completing the development process, a self-evaluation was done to explore that whether the developed UI fulfills the claimed features or not. Next, few recommendations were proposed to improve the user experience of the project manager as the end user.

Figure 2 illustrates and summarizes the structure of the thesis.

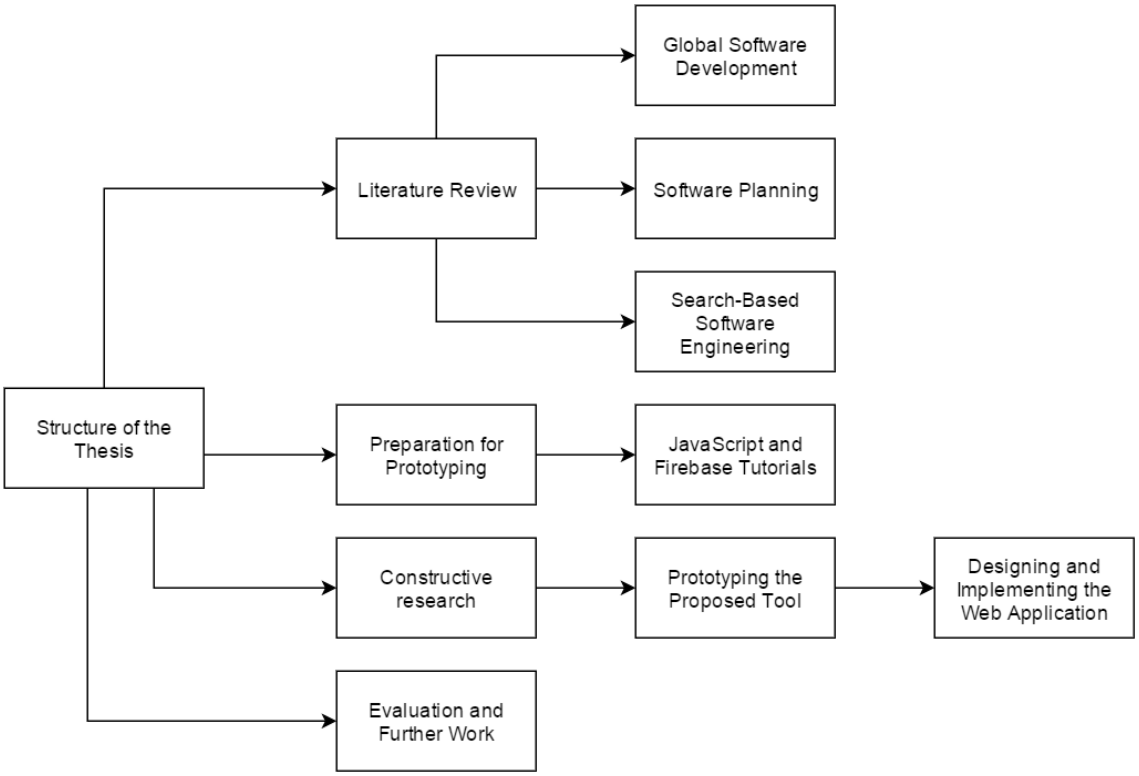


Figure 2. Structure of the thesis

The next chapter focuses on the background areas including GSD, software planning and SBSE. The third chapter presents the research set-up including the research of Vathsavayi

on search-based planning of GSD projects by proposing a tool support. The fourth chapter focuses on the features, requirements, and UI of the developed application. The fifth chapter presents the development process, tools and technologies, database, and software testing. The sixth chapter includes the evaluation of the developed UI as well as some proposed improvements. The last chapter finally presents the conclusion of the thesis.

2. BACKGROUND

2.1 Global Software Development

2.1.1 Introduction to Global Software Development

More than a decade ago, many software companies started to outsource the work to remotely located software development sites in order to access to more skilled resources (teams) and lower costs [7]. Global software development (GSD) is defined as a software development that is distributed in multiple sites (software development centers) that are possibly located in different countries and even in different continents. Teams involved in a GSD project collaborate with each other to perform the activities of the project [8]. In other words, a GSD project involves a set of teams that work together to obtain the same goals in the same project, however the teams are in different geographical locations [9].

There are several benefits that motivate organizations to practice GSD, as they are listed in the following.

- GSD takes advantage of the global pool of software development resources wherever they are located [7] [10].
- GSD takes advantage of business benefits of getting geographically closer to the market and customers in order to localize and customize more efficiently [7] [10].
- GSD takes advantage of the possibility of foundation of virtual corporations and teams based on current market opportunities [7]. This means that a group of individuals performing a virtual team are selected based on current market opportunities to work together on a GSD project across the time, space, and organizational boundaries.
- GSD improves the time-to-market as a result of having time-zone differences. In a GSD project involving developers in different time-zones, the organization might increase the number of working hours in a day. As a result, the cycling time decreases, and the productivity increases. This is referred to as follow-the-sun development which is considered as a potential advantage of practicing GSD. [7] [10]
- One of the most important advantages that is expected to achieve in a GSD project is the significant reduction in development cost as a result of considerable differences between development cost in developed and developing countries. [10] [11]

- It is expected that GSD gets benefits from increasing innovation and sharing best practices among teams, since teams and individuals acting in a GSD project are from various backgrounds. [10]
- GSD is expected to improve cross-site modularization of development. In a GSD project, the work should be broken up into well-defined modules. Then, the modules are assigned to the teams to be developed in parallel. The overall development process is expected to get benefit from such a modularization across the development sites. [10]

GSD projects are typically multisite, multicultural, and geographically distributed. Hence, engineers, project managers, executives, and all people involved in a GSD project experience different types of cultural, social, and technical challenges. Most of the people working in a GSD project experience misunderstandings and difficulties that are related to GSD challenges. Furthermore, there is evidence that multisite development takes much longer than collocated development due to communication and coordination overhead. Realizing the benefits of GSD projects is not completely straightforward due to several typical issues and difficulties that are explained and discussed in more detail in the next section. [7]

2.1.2 Typical Issues of Global Software Development

In the following, the typical issues that GSD projects most likely face are explained and discussed.

1. **Cultural issues:** Normally in a GSD project several people from different cultures need to be in close collaboration to meet the goals of the project. Cultures differ on many different aspects such as sense of time, style of communication, attitude towards hierarchy, and the need for structure. These cultural differences may lead to misunderstandings mostly when people do not know each other well. As an example, in some cultures communication tends to be extremely direct while it is not appreciated in some other cultures and can even considered to be rude. All of these cultural differences have potential to increase communication problems. [7]

   Culture can have a significant influence on how different people interpret a particular situation as well as how they respond to that situation. There is a socio-cultural distance in GSD that specifies how well a person can understand the values and the norms of another person. In addition to the national cultural background, the organizational culture of the people should be taken into account. There can be a low socio-cultural distance between two people having different nationalities and cultural backgrounds, but sharing the same organizational culture. Meanwhile, it can be a high socio-cultural distance between two people having the same nationality and culture, but working in two different companies with

different organizational backgrounds. Definitely, the geographical distance may increase the cultural differences. However, there can be huge cultural differences in low geographical distance. Meanwhile, huge geographical distance does not necessarily result in huge cultural differences. [12]

2. **Communication issues:** Software development needs a lot of communication especially in its early stages [7]. People involved in a software project need to communicate together to understand efficiently the problem statement, project requirements, responsibilities and tasks, and project status updating. People involved in a GSD project do not often have the opportunity to frequent and face-to-face communication due to geographical distances between the development sites [13]. In a study of engineering organizations, Tom Allen [14] states that the frequency of communication among engineers decreases proportionally to distance. In addition, he notes that when the offices of engineers are located more than 30 meters far from each other, the frequency of communication decreases the same as the condition in which the distance between offices is several miles. Kraut et a1 [15] got the same results for scientists, meaning that the rate of spontaneous collaboration between scientists is a function of distance between scientists' offices.

   Software projects usually need both formal and informal types of communication. However, the communication is basically more difficult in a GSD project. Firstly, a software project needs formal type of communication to perform some of the vital tasks of the project such as determining the responsibilities, updating the status of the project, escalating the project issues and so on. Secondly, a software project needs to informal type of communication. In a GSD project, developers working in different sites possibly have very little opportunity to have informal and spontaneous conversation across the sites. Hence, informal communications can help people working in different sites of a same GSD project to stay aware about several aspects that are listed in the following.
   - What is going on around them?
   - What are other people working on?
   - What states different parts of the project in other sites are in?
   - Who has skill and experience in what area and what level?
   - Many other information that helps developers to work together more efficiently.

   The lack of adequate formal and informal types of communication in a GSD project leads to misalignment and rework as well as increase in cost and duration of the GSD project. [7]
3. **Task assignment issues:** Making decision on how to divide work among several sites of a GSD project is a critical task. In order to assign a task to a project site, the available resources of the site, their skills as well as the corresponding level

of skills need to be taken into account. An ideal task assignment makes different sites of the GSD project to work as independently as possible. So, the need for extra communication between sites decreases dramatically. [7]

Moreover, the project manager, who is responsible for assigning tasks to the teams, needs to pay too much attention to both duration and cost of the GSD project. There are many different solutions in assigning tasks to the teams. Each solution can affect cost and duration of the GSD project differently. The project manager technically cannot evaluate all of the possible solutions to find out the optimized solutions (a set of non-dominated solutions in a tradeoff between cost and duration of the GSD project). As a result, task assignment is a very difficult task in planning a GSD project. [1]

4. **Technical issues:** A common technical issue in a GSD project is using of incompatible data formats or different versions of the same tools in different sites of a GSD project [7]. Another typical technical issue is related to networks that make the sites in different geographical locations communicate and coordinate together and transmit the critical data [7]. As an example, such networks might be slow and unreliable. This can cause issues in communication between sites in different geographical locations. The other technical issue in a GSD project is that there is a possibility that the adopted tools and technologies in the GSD project are upgraded or even changed during the project [7]. Such changes in tools and technologies may result in problems in some remotely-located sites, since the people working on those sites might not be familiar with new tools and technologies. So, it takes time for training people and it causes delay in delivery of product. In general, technical issues in a GSD project are mostly originated from differences in infrastructure in different sites [16]. These differences are generally related to development environments, test labs, version management systems, and network connectivity [16].

5. **Knowledge management issues:** In a GSD project, knowledge management is a challenging task, but it is a key factor in the survival of the project [17]. A common challenge in knowledge management in a GSD project is related to efficiently sharing information among people involved in the project. As an example, the project manager might fail to share equally the significant information about customers to different sites located in different geographical locations. Another example of failure in knowledge sharing is when teams are informed inadequately about the current status of the project, i.e., a development site is not informed adequately about what is going on at the other development sites. A further example is poor documentation which can result in inadequate sharing of knowledge and information between sites, and consequently cooperation between different

sites cannot be effective. Resistance to documentation is common among developers, while documentation is a vital activity particularly in GSD projects. In addition to documentation, updating and revising the documentation is necessary in GSD projects. [7]

6. **Project and process management issues:** In a GSD project, synchronization between teams is problematic. For example, the development team at one site may need to be synchronized with the test team at another site. In order to develop synchronization among teams, milestones, entry and exit criteria should be defined commonly between sites. Concurrent engineering principles seem to be helpful in theory to synchronize teams. Concurrent engineering is a methodology for designing and developing products, in which different tasks run simultaneously rather than consecutively in order to decrease the time of product development [18]. However, concurrent engineering principles cannot be practically applied in a GSD project. The reason is unstable requirements and specifications of a GSD project as well as inadequate access to collaborative tools and formal communications. Another way to synchronize teams involved in a GSD project is the practice of risk management, however the effects of cultural differences and different attitudes need to be taken into account. [7]

2.1.3 Concept of Distance in Global Software Development

In a GSD project, there are several different types of distance that are listed in the following.

- Geographical distances between teams prevent people from different teams to have frequent and face-to-face communication [19].
- Time-zone differences limit real-time communication between teams located far from each other [19].
- Cultural characteristics differences result in communication problems due to misunderstanding between teams with different cultural backgrounds [19].
- Linguistic distances due to lack of a common native language result in further barriers to communication between teams [20].

Considering the above mentioned different aspects of distance in a distributed work, it can be concluded that we easily think about the concept of distance in a slightly limited way. As a matter of fact, the geographical distance need not be necessarily huge to have a significant effect. For example, locating on another building, or on a different floor of the same building, or even on the other end of a long corridor can significantly reduce the communication. It reflects the fact that the solutions that might help GSD to be done more effectively can also revise the same issues in a same-site project. As a conclusion to the concept of distance, all different aspects of distance should be taken into account

in GSD projects. Distance has a wider concept compared to what we have initially assumed. Hence, different aspects of distance should be analyzed and surveyed carefully as much as possible to resolve some of the typical issues of GSD projects.

2.2 Project Planning

2.2.1 Definitions of Project Planning and Project Management

Project planning is a significant part of project management, and project management itself is one of the most important parts of software projects that plays a significant role in the success or failure of the software project. Failure of a software project happens in many cases just because the project manager does not know the actual status of the project and consequently cannot control the project especially when the project is too large. Such a misalignment happens mostly in software projects, since software is more difficult to visualize than hardware or any other physical products. In many software projects, it is common to face difficulty in meeting the deadlines and delivering the product with the promised features and within the budget. [21]

PMBOK [22] defines the project integration management as all processes required in order to guarantee that all elements of the project are appropriately coordinated. Following categorization of processes gives an overview of the process of project integration management.

1. Creation of project plan that includes gathering all inputs from other planning processes (that will be discussed later in the current section) and putting them into a coherent, consistent, and formal document that needs to be approved.
2. Execution of project plan that means performing the project following the activities and schedules specified in the project plan.
3. Overall change control that means updating the project plan during the progress of the project. Updates are any modifications to the content of the project plan.

Complex projects most likely fail without a project plan. Two proverbs that emphasize the significance of the project planning are listed in the following [23].

- "Failing to plan is planning to fail."
- "The primary benefit of not planning is that failure will then come as a complete surprise rather than being preceded by periods of worry and depression."

Before starting a software project, project planning should be done as a vital necessity. Software projects cannot be managed efficiently without having a realistic and appropriate project plan. For a software developer, whose goal is the satisfaction of customers' needs, plan is a complete, consistent, and coherent expression of the stakeholders' requirements. [24]

PMBOK defines the project plan as a consistent, coherent, and formally-approved document that is used to guide and manage the execution and control of the project [22]. Project plan is a dynamic document meaning that it is expected to be changed during the project as new information becomes available. Project plan is used for multiple purposes that are listed in the following.

- Guiding the execution of the project
- Documenting the assumptions of project planning
- Documenting the chosen alternatives of project planning
- Facilitating the communications among stakeholders
- Providing a baseline for measuring the progress and control of the project

2.2.2 Development of Project Plan

The development of a project plan is an iterative process. PMBOK divides the process of developing a project plan into three categories including inputs, tools and techniques, and outputs. They are explained separately in the following. [22]

Inputs for developing a project plan:

- Constraints: project mangers' options are limited by factors that are considered to be constraints of the project. For example, a predefined deadline or budget is a constraint that limits the available options.
- Assumptions: in order to make a plan, assumptions are factors that are assumed to be realistic. Assumptions generally add a degree of risk to a project. As an example, if a significant date in a project is uncertain, the project team may need to assume a specific date.
- Organizational policies: organizations may have formal and informal policies that can influence different parts of the project. The effect of such policies should be taken into account in a project plan.
- Historical information: examples of available historical information are records of effort estimation and past project performance. In a project plan, such historical information is used for the purpose of learning and consulting. So, such historical information should be available during the development of project plan to assist with evaluating the alternatives and confirming the assumptions.
- Other planning outputs: all of the outputs (results) of planning process of the other projects may be beneficial for a new project plan. For example, some base documents such as work breakdown structures may be useful.

Tools and techniques for developing a project plan:

- Methodology of project planning: any structured approach that assists the project manager in developing a project plan is referred to as methodology of project planning. The methodology may be simple such as standard forms and templates or it may be more complex. Most of the methodologies utilize project management software and facilitated start-up meetings.
- Stakeholders' skills and knowledge: each stakeholder may have practical skills and knowledge that might be helpful in development of a project plan. So, the project manager is assumed to be responsible for developing an appropriate environment in which all stakeholders can have the opportunity of contributing their skills and knowledge.
- Project management information system: as it was mentioned earlier, outputs from the other project management process may be considered as the input to a new project plan development process. A project management information system (PMIS) is any kinds of tools or techniques that can be used to collect, integrate, and publish the outputs from the other project management processes.

Outputs from developing a project plan:

- Project plan: to recap, project plan is a formally-approved document that guides the project manager in managing and controlling the execution of the project. Such a document is expected to be changed over the time as new information will be available.
- Supporting documents: in addition to the project plan document, the supporting documents are another outputs from developing a project plan. Supporting documents can be any of the following materials.
    1. Outputs from other planning process that were not included in the project plan
    2. Extra documents created during the project plan such as constraints and assumptions that were not known in advance
    3. Technical documents such as requirements, specifications, and design documents
    4. Documentation of standards related to the project plan

  This kinds of supporting documents should be organized to be used during the project execution.

2.2.3 Planning of Global Software Development Projects

When planning a GSD project, the project manager is frequently involved in many decision-making situations such as which teams are more suitable for assigning work to, or how to distribute work optimally among teams in a tradeoff between cost and duration of

the project. There are many alternative solutions, and each solution has a different influence on the cost and duration of the GSD project. As a result, finding an optimal solution by the project manager without any automated support seems to be a very difficult task. [1]

As it was mentioned earlier in 2.1.3, there are different types of distance such as geographical, time-zone, cultural differences and linguistic distances between teams participating in a GSD project. Due to these distances, team communication, coordination, and control are difficult tasks. As a result, it is not an easy task for the project manager to recognize the cost and duration advantages of a GSD project. To overcome this difficulty, the GSD project needs to be planned as efficient as possible. [1]

When planning the GSD project, the project manager divides the work into multiple work packages and tasks, schedules them, and assigns them to the available teams. There can be many different possible work divisions, work allocations, and schedules. Furthermore, there can be an exponential growth in number of applicable team combinations when the number of available teams increases [1]. Exponential growth [25] is a model for growth of a quantity, in which the rate of growth is proportional to the present amount. The equation representing the exponential growth is $Y = Xr^{kt}$, where X is the original amount present at time $t = 0$, and k is a positive number representing the rate of growth. When there are many possible team combinations, the project manager obviously cannot evaluate all of them. Hence, an automated tool support for planning the GSD project seems to be a helpful idea. Vathsavayi proposes an automated tool support to assist the project manager in planning a GSD project. The tool is explained and discussed in detail in the research set-up chapter (next chapter).

2.3 Search-Based Software Engineering

2.3.1 Introduction to Search-Based Software Engineering

Search-based software engineering (SBSE) involves the application of search-based optimization techniques in the area of software engineering by formulating a software engineering task as a search problem by defining the following two items.

1. An appropriate representation of the candidate solutions.
2. A suitable function that can differentiate between the solutions to realize which one is the better solution.

The representation of candidate solutions to the problem specifies the search space in which the search for the solution occurs. In order to assist the search-based optimization, a specific function is required to be defined. Such a function needs to have the ability of determining the better solution of two candidates in order to go through the search space and progress by differentiating between solutions. [26] [27]

The most popular algorithms practiced in SBSE have been as following [28].

- Genetic Algorithms
- Genetic Programming
- Simulated Annealing
- Hill Climbing

In addition to the above mentioned algorithms, there are many other optimization techniques such as ant colony optimization (ACO) that have been already applied in SBSE [29]. However, the basic idea of search-based formulation is the same regardless of the applied algorithm, i.e., the problem is considered as a search among a large space of the candidate solutions [27].

2.3.2 Application Areas of Search-Based Software Engineering

Search-based optimization has been applied for many years in several engineering areas such as mechanical, materials, chemical, electric, and electronic engineering. Also, software engineering has recently started to practice search-based optimization. SBSE has a great potential to be applied in software engineering. It is applicable to highly-constrained problems involving many potentially competing and conflicting objectives that need to have an automated approach. SBSE is potential to be an evolution of software development into a more mature engineering discipline. SBSE has a wide diversity of application such as designs, test cases, and behavioral and quality models. [27]

Currently, SBSE has been successfully practiced to construct the following software artifacts.

- Making optimal test cases [30]
- Clustering of modules and heaps [31]
- Requirements sets [32]
- Project management [33]
- Refactoring [34]

Furthermore, a very interesting and particular application area of SBSE is in software planning. SBSE can be used to schedule and staff a software project. Vathsavayi has conducted a research [1] on proposing an automated tool that guides the project manager in planning a GSD project. Vathsavayi uses multi-objective genetic algorithms [2] to find near-optimal solutions to team selection as well as task allocation. The next chapter (research set-up) concentrates on the proposal of Vathsavayi.

3. RESEARCH SET-UP

3.1 Research Process of the Thesis

In this thesis, the research of Vathsavayi [1] was studied as a starting point. It includes a GSD work distribution model and some preliminary ideas for developing an automated tool support that assists the project manager in planning a GSD project. Hereafter, the automated tool support for GSD project planning proposed by Vathsavayi is called SB-GSD tool. To comprehend the research of Vathsavayi in detail, some literature study was conducted about the background topics such as GSD and its typical challenges, concept of distance in the context of GSD, project management and project planning of GSD, and the application of search-based optimization in software engineering. Also, few research was done about whether or not some of the popular project management software products such as Microsoft Project support GSD projects. Moreover, some literature review was conducted about the existing automated supports for planning GSD projects. Finally, the research of Vathsavayi was evaluated in some limited aspects. Firstly, a discussion was presented about how the proposed GSD model alongside the SB-GSD tool addresses some of the common issues of GSD projects. Moreover, few recommendations were proposed to estimate efficiently the communication distances between the participating teams.

After the literature study, the constructive research part of the thesis was started. A suitable and practical user interface (UI) that is assumed to be used by the project manager was designed and implemented for the SB-GSD tool. The reason for developing a new UI for the SB-GSD tool was that the existing UI initially developed by Vathsavayi was more suitable for researchers to understand the basic idea behind the SB-GSD tool. Hence, there was a need for developing a new concrete UI that can be used by project managers when planning the GSD project.

The developed UI for the SB-GSD tool is a simple web application that allows the project manager to gather data about the GSD project as well as the available GSD teams. The data is stored into a database and can be retrieved and updated by the project manager at any time as new information about the GSD project and the available teams is provided. In order to design such an application, all required input fields were extracted directly from the GSD work distribution model [1] presented in the research of Vathsavayi. The extracted input fields were evaluated by the research team. To develop the web application, some of the web development technologies such as HTML, CSS, and JavaScript were utilized. Also, a Firebase database was used to store the data about the GSD project and the participating teams. Firebase is a cloud service that provides a real-time database. The UI was developed during multiple phases in an iterative and incremental method of development based on regular feedback from the research group. Finally, the developed

UI was evaluated by conducting a self-evaluation. Next, some further work was proposed to improve the UI so that it can provide a better user experience for the project manager as the end user.

To conclude this section, a simple web application (a concrete UI for the SB-GSD tool) was developed as a result of four phases of iterative prototyping. The developed application is assumed to be used by the project manager to gather data about the GSD project and the participating teams. The data is managed by a Firebase database so that it can be stored, retrieved, and updated by the project manager. Finally, the data can be used by the SB-GSD tool to plan the GSD project.

3.2 Search-Based Project Planning in GSD

Vathsavayi in his research [1] proposes the SB-GSD tool that utilizes multi-objective genetic algorithms (GAs) [2] to find the optimal work allocations and team combinations in a GSD project. The tool assists the project manager by analyzing how different work allocations and team combinations can influence the duration and cost of the project. This is done by providing the project manager with a palette of solutions in time and cost scale. Then, the project manager can explore all of the provided solutions in the palette and select the most efficient solution according to the required duration and cost of the GSD project. The number of possible combinations can grow exponentially. Hence, to find the near-optimal solutions, the automated tool support is based on applying meta-heuristic search algorithms instead of deterministic search algorithms. The SB-GSD tool is supposed to get data about the GSD project and the available teams as the required inputs. Next, the tool uses GAs to find the optimal solutions for scheduling and allocating the work to the teams in a tradeoff between cost and time. Finally, the project manager explores the proposed solutions, and selects the most suitable one.

In summary, the main contributions of the proposal of Vathsavayi [1] are listed in the following.

- Assisting the project manager in planning the GSD project by developing an automated planning tool support that explores the optimal work allocations as well as the duration and cost of each work allocation.
- Applying multi-objective genetic algorithms in order to find near-optimal work allocations.

Also, the main contribution of this thesis is the development of a simple and concrete UI for the SB-GSD tool. The UI is used by the project manager to add, delete, update, store, and retrieve data about the GSD project and the participating GSD teams. The data stored in the database is considered as the required inputs to the SB-GSD tool proposed by Vathsavayi.

3.3 GSD Work Distribution Model

In the proposal of Vathsavayi, the GSD work distribution model [1] includes the following data.

- Data about the available teams
- Data about the system to be developed in a GSD project

The data is consequently used as the inputs to the SB-GSD tool. Figure 3 illustrates the class diagram of the GSD work distribution model.

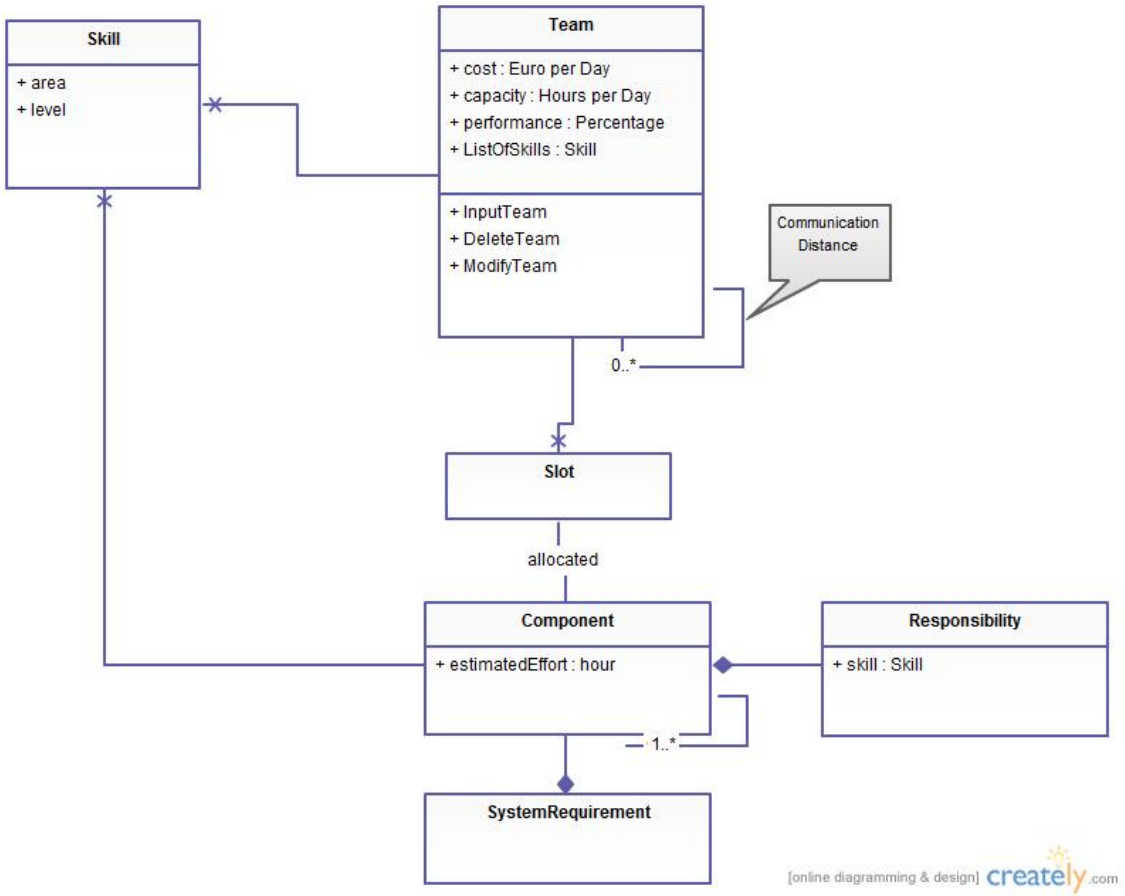


Figure 3. Class Diagram of the GSD Work Distribution Model. Reproduced with permission of the author. [1]

3.3.1 Concept of the System to be Developed in the Model

In the GSD work distribution model, the system to be developed is defined in terms of logical components. Each component is a work package consisting of a set of responsibilities. Each responsibility is a basic functional task required to accomplish the functionality of the system. Each responsibility needs a different skill. Hence, each component is determined in terms of a set of skills required to develop that component. Initial functional decomposition of the system determines the major components required to develop the

functionality of the system. In addition, an estimated effort is specified for each component which is the estimated total number of working hours required for developing that component. [1]

Each component has two kinds of relationships with the other components as dependency and precedence relationships. [1]

1. Dependency relationships are determined by initial functional decomposition of the system. Dependency relationship happens when a component needs a particular service from another component. As an example, when we determine that component A depends to component B, this means that some information from component B is needed when developing the component A. As a result, some communication between teams developing the components A and B is required.
2. Precedence relationships specify the preferred development order of the components due to strong dependency between those components. Precedence relationships cannot be specified by using any automated support. Hence, the architect is assumed to be responsible for determining the precedence relationships between the components. Precedence relationships allow the project manager to express some extra information regarding the development order of the components.

3.3.2 Concept of the Participating Teams in the Model

In the GSD work distribution model, a team is differentiated from the others by a set of characteristics that are listed in the following. [1]

- Cost of the team per day
- Team's capacity which is determined by the number of hours a team can spend per day
- Team's performance in terms of percentage
- Team's skills and their corresponding levels of experience: several developers are involved in a team, and each developer has multiple skills with their corresponding levels of experience. Hence, a team is defined in terms of a set of skills and their corresponding levels of experience.

In addition to the mentioned characteristics, the following concepts are associated to the concept of a team in the GSD work distribution model. [1]

- Slot: a team is associated with several slots, and each slot is the position of a component in the team's development order.
- Communication Distance: each team has a certain distance from other teams which is referred to as communication distance. Communication distance specifies how efficiently two teams can communicate together in a GSD project.

3.4 Examples of Search-Based Project Planning in GSD

Vathsavayi uses the e-home project as an example to show the application of his proposed tool support. Figure 4 illustrates the functional decomposition of the e-home system. Also, it shows the logical components, the relationships between them, as well as the estimated effort and the required skills for each component.

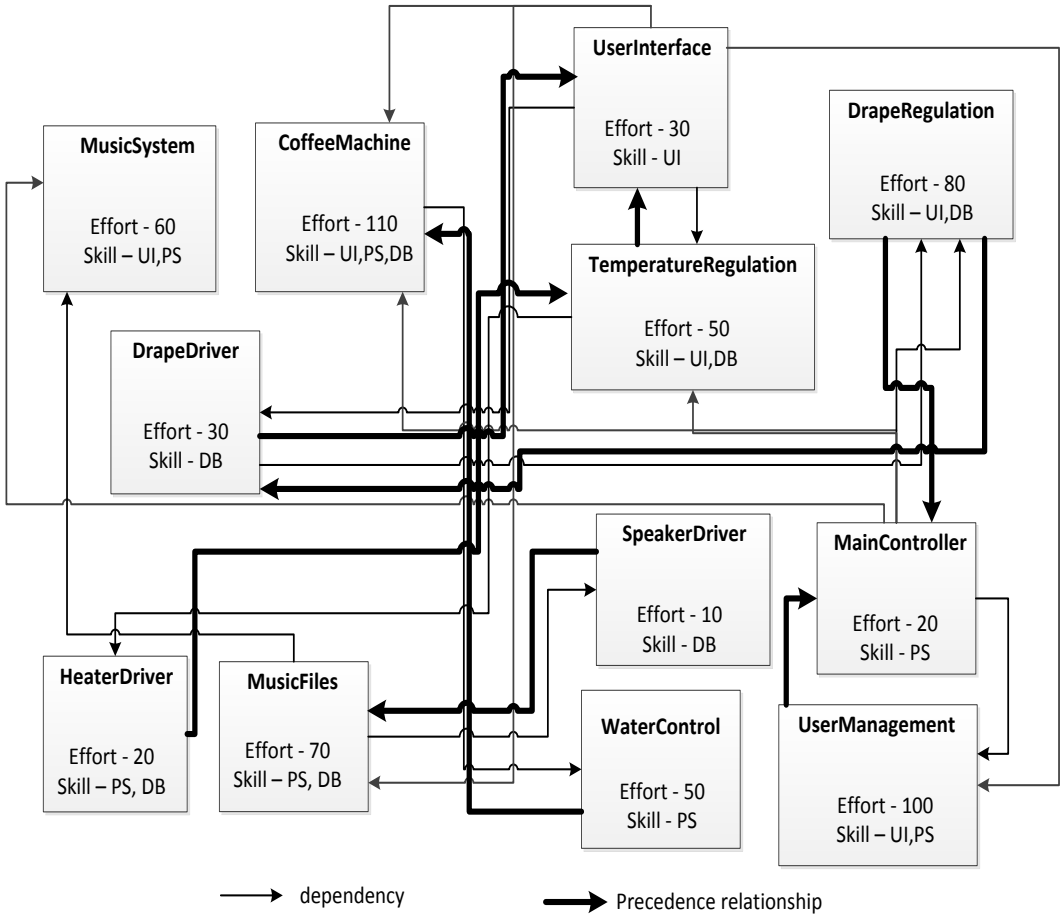


Figure 4. Functional decomposition of the e-home project by Sri Vathsavayi, 2014. Used with permission.

As the first example of decision-making, it is assumed that the project manager is exploring whether or not it is beneficial to use low-cost remote teams instead of teams located in the same site. There are two different work distributions for the project manager developing the e-home project. In the first work distribution, teams are located in the same site, for example in Los Angeles. In the second work distribution, teams are located in different sites, for example in Los Angeles, New Delhi, and Berlin. Figure 5 and Figure 6 show these two work distributions. The level of experience is assumed to be the same in all available teams. The SB-GSD tool gets data about the available teams and the e-home project as the required inputs and consequently produces a Pareto front as the output. At this point, a Pareto front is a palette of non-dominated solutions (work allocations)

in time and cost scale. Figure 7 illustrates the Pareto front related to the mentioned two work distributions. In the figure, the small circles are related to the first work distribution (teams in the same site), while the bigger circles are related to the second work distribution (teams in different sites).

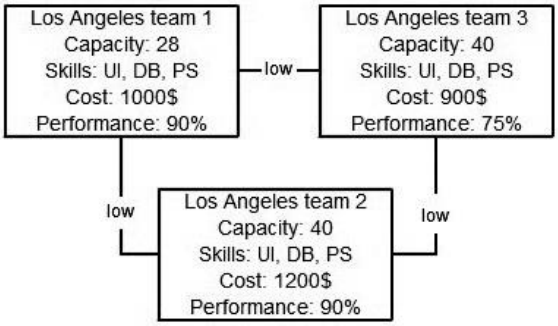


Figure 5. Teams in the same site as the first work distribution

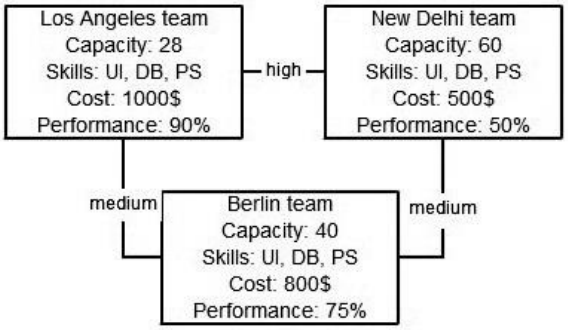


Figure 6. Teams in different sites as the second work distribution

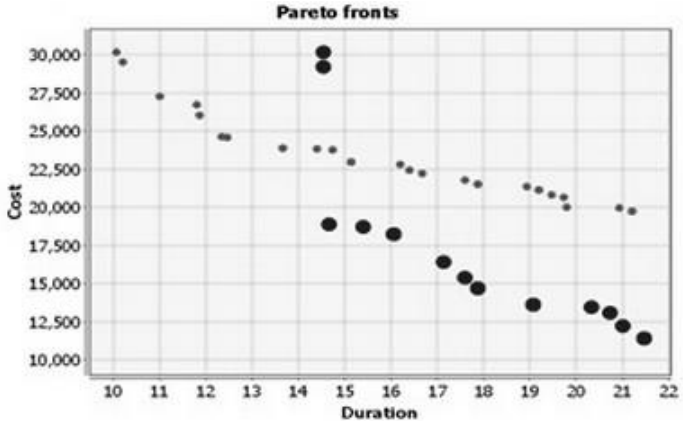


Figure 7. Non-dominated solutions by Sri Vathsavayi, 2014. Used with permission.

The project manager can explore the above Pareto front to analyze which work distribution is more suitable for the e-home project. As it can be seen from Figure 7, the project manager can conclude the following results from the Pareto front. [1]

- When the e-home project is developed by the same-site teams, the duration required to accomplish the work is shorter than when the e-home project is developed by the teams in different sites.
- When the e-home project is developed by the teams in different sites, the cost is less than when it is developed by the teams in the same site.

However, it is not always possible to reduce the cost by sending work to low-cost remote teams, since more time for communication between teams is required. Hence, when sending work to the low-cost remote teams, the required amount of communication between teams as well as the number of teams should be taken into account by the project manager.

For the next example of the application of the SB-GSD tool, it is assumed that the project manager is exploring which teams among the available teams are more suitable for participating in the GSD project. As an example, it is assumed that the Los Angeles team needs to select one team among Berlin and New Delhi team as the available teams to develop the e-home project. At this point, the SB-GSD tool guides the project manager in exploring both advantages and disadvantages of involving each teams (Berlin and New Delhi) by providing the project manager a helpful Pareto front. Furthermore, if the project manager is in doubt about some of the input parameters such as the communication distances between the teams, the SB-GSD tool allows him/her to examine the impact of the variation of that parameter on the set of non-dominated solutions presented in the Pareto front.

In conclusion, the SB-GSD tool gets the input data about the GSD project and the available teams from the project manager, and then provides a Pareto front as the output presenting a set of near-optimal non-dominated solutions. Next, the project manager can choose the most appropriate solution in a tradeoff between the duration and cost of the GSD project.

3.5 Evaluation of the GSD Model and the SB-GSD Tool

This section explains how the GSD work distribution model and the SB-GSD tool address some of the typical issues of a GSD project.

Cultural and communication issues: The GSD work distribution model represented in 3.3 addresses the cultural and the communication issues of a GSD project represented in 2.1.2 by considering a meaningful communication distance between each pair of teams. In the model, each team has a certain communication distance from other teams that should be estimated by the project manager. Next, the SB-GSD tool takes the values of the communication distances into consideration when exploring the optimal work distributions. The GSD model addresses the cultural issues by considering the cultural differences between each pair of teams as one of the factors that should be evaluated by the project manager when estimating the communication distances. Furthermore, the GSD model addresses the communication issues by considering the time-zone differences as

well as the geographical distances between teams as the other factors required to be evaluated by the project manager.

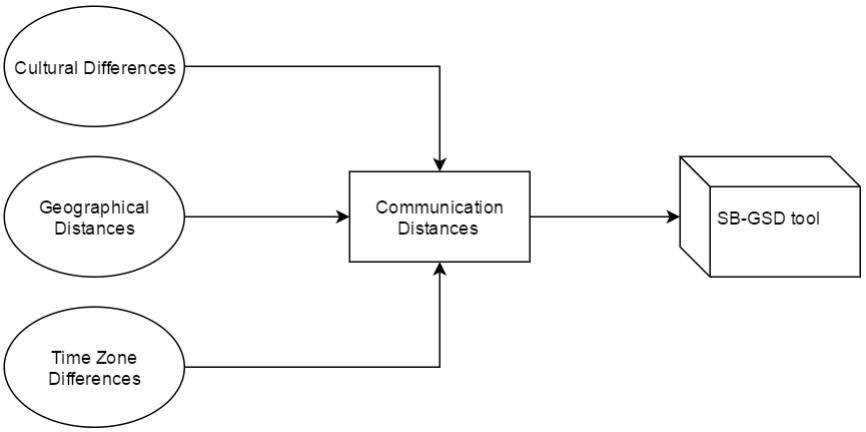


Figure 8. Addressing cultural and communication issues

Task assignment issues: The proposed SB-GSD tool addresses the task assignment issues of a GSD project represented in 2.1.2. As it was mentioned earlier in 3.2, the SB-GSD tool utilizes multi-objective genetic algorithms to explore how different work allocations influence the cost and the duration of the GSD project. Next, the SB-GSD tool offers the project manager a palette of optimal work allocations. Hence, the SB-GSD tool obviously supports the project manager in dealing with the task assignment issues.

3.6 Further Work for Estimating the Communication Distances

In this section, some further work is proposed in order to enrich the concept of communication distance between teams.

Considering a linguistic distance between teams: The GSD model is proposed to consider a linguistic distance between each pair of teams. The linguistic distances should be evaluated by the project manager when estimating the communication distances. Both factors of the native language and the level of English proficiency can influence the linguistic distance between a pair of teams. As an example to prove the significance of the native languages of the teams, there is almost no linguistic distance between a French speaking team in Switzerland and a native team from France. Also, as an example to prove the significance of the level of English proficiency of the teams, there is a high linguistic distance between a French team with no proper level of English and a native British team. So, the French and British teams cannot communicate efficiently despite the fact that they have low geographical distance and no time-zone distance. Hence, considering a linguistic distance between teams can modify the concept of communication distance presented in the GSD work distribution model. Figure 9 represents taking the linguistic distance into consideration in the SB-GSD tool.

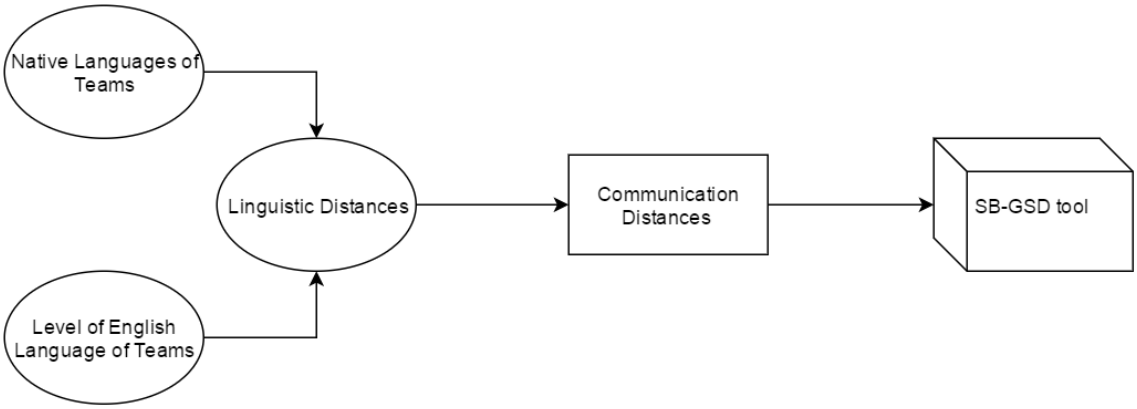


Figure 9. Linguistic distances

Considering some assessment criteria for geographical distance: In the GSD model, it has not yet determined how the project manager should evaluate the geographical distance between two teams. In addition to physical distance between two teams, locating within a same country or different countries can influence the communication distance between teams. For example, it is possible that there is only 10 kilometers distance between two cities, while they are located in two different countries with different languages and different cultures. In contrast, two teams can have more than 1500 kilometers distance but still located within a same country with a same culture, language, and time zone. Furthermore, the availability of easy and fast transportation between the cities of teams can influence the distance between them. Hence, it is proposed that the project manager proceeds with the evaluation of geographical distance based on some specified criteria. Figure 10 shows the preliminary assessment criteria proposed for estimating the geographical distance.

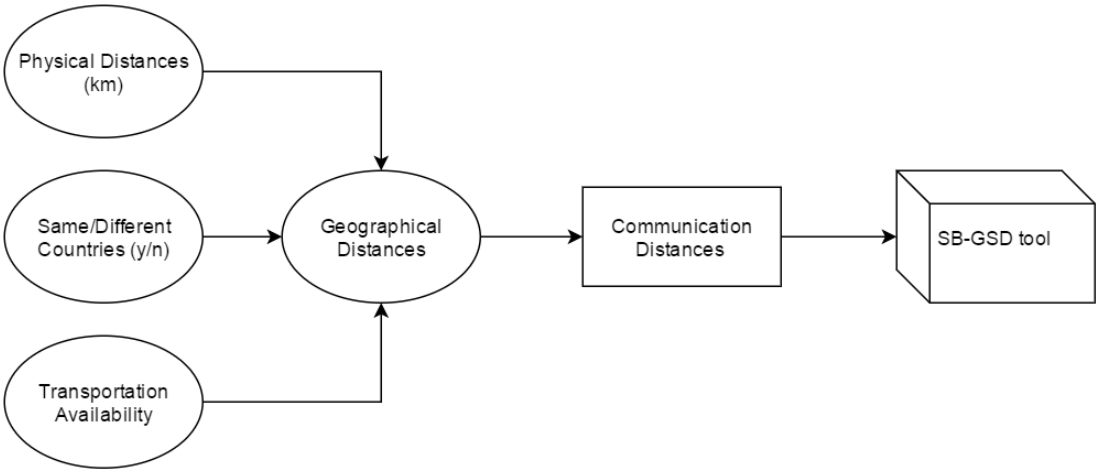


Figure 10. Geographical distance

Considering some assessment criteria for cultural differences: In the GSD model, it has not yet determined how the project manager should evaluate the cultural differences between two teams. First of all, the culture of a team is not necessarily originated from the native culture of the country in which the team is located. In many countries such as

US, UK, Australia, and Singapore foreigners and locals have been integrated too much. In such countries, the culture of a team involving both locals and foreigners (possibly with different nationalities) is expected to be quite different from the native culture of the country. Hence, in order to evaluate the cultural differences between two teams, the cultures and the nationalities of the individuals should be explored instead of the culture of the country in which the team is located. Secondly, in many countries the native people are extremely multicultural and multiethnic. For example, Brazil and India have a vast diversity of cultural differences and ethnic groups. As a result of such cultural differences, two teams from India can have a significant cultural differences. Hence, cultural diversity of native people from the same country should not be neglected. In conclusion, some specified assessment criteria can assist the project manager in evaluating the cultural differences between teams. Figure 11 represents some of the items that should be taken into consideration when evaluating the cultural differences between GSD teams.

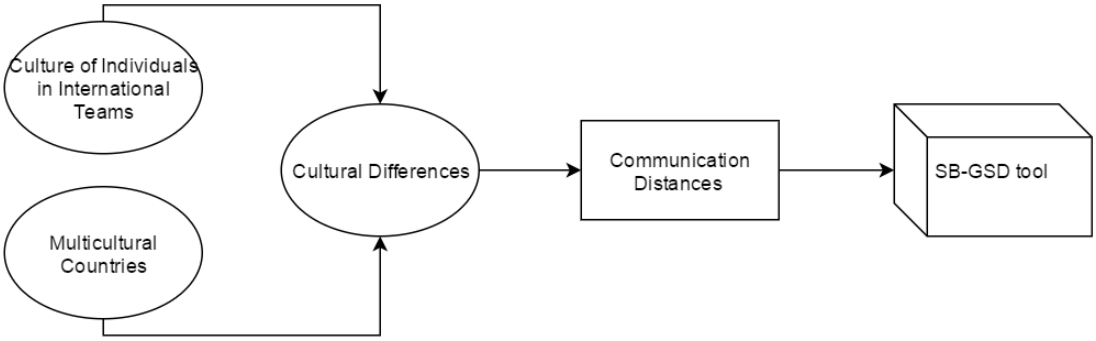


Figure 11. Cultural differences

4. APPLICATION

4.1 Developed Application and its Main Features

The developed application in the thesis is a concrete user interface (UI) used by the project manager to plan a GSD project based on the SB-GSD tool. The chosen name for the developed application is UI-GSD. The UI-GSD application is the second UI that has been already developed for the SB-GSD tool. The first UI was developed by Vathsavayi in order to be used by the researchers to understand the whole idea behind the proposed tool. Hence, that UI was not suitable for the project manager to interact with when planning the GSD project. The second UI was developed by me, and it aims to enable the project manager to add, delete, update, and retrieve data about GSD projects and available teams. Data is stored in a database. The stored data is assumed to be used by the SB-GSD tool to plan the GSD project automatically.

The main features of the developed application are listed in the following.

1. The application allows the project manager to gather data about the available teams possibly participating in the GSD project.
2. After adding the available teams, the application allows the project manager to estimate the communication distances between teams.
3. The application allows the project manager to gather data about the components of the GSD project.
4. After adding the components, the application allows the project manager to specify the dependency and precedence relationships between components.
5. All of the data provided by the project manager is stored in a database. The SB-GSD tool can use the data as the input data to present a set of non-dominated work distribution solutions in a tradeoff between the cost and duration of the GSD project. Then, the project manager can explore the solutions to get to know which teams among the available teams are more suitable for participating in the GSD project, and how to assign work to the chosen teams considering the cost and duration of the GSD project.
6. The application allows the project manager to retrieve, update, and delete the data stored in the database at any time as new information about the GSD project and the available teams is provided during the project.

4.2 Requirements of the Application

Functional and non-functional requirements of the developed application are listed in the following table.

Table 1. Functional and non-functional requirements

List of Requirements	**Priority**	**Functional**	**Non-functional**
The application is a simple, clear and user-friendly UI.	high		✓
The application runs in web browsers.	high		✓
The application allows the project manager to input data about a new team and store it into the database.	high	✓	
The application allows the project manager to update the data of a team stored in the database.	high	✓	
The application allows the project manager to delete a team from the database.	medium	✓	
The application allows the project manager to input data about the communication distances between teams and store them into the database.	high	✓	
The application allows the project manager to update the data of the communication distances stored in the database.	high	✓	

The application allows the project manager to input data about a new component and store it into database.	high	✓	
The application allows the project manager to update the data of a component stored in the database.	high	✓	
The application allows the project manager to delete a component from the database.	medium	✓	
The application allows the project manager to input data about precedence/dependency relationships and store them into the database.	high	✓	
The application allows the project manager to update the data of the precedence/dependency relationships stored in the database.	high	✓	

4.3 Designing the User Interface

Some of the principles that have been followed to design the UI of the developed application are explained in the following.

Simplicity: The designed UI aims to make the tasks on teams and components simple and easy to do.

Consistency: The designed UI aims to provide consistency in the used language, elements, and layouts throughout the application.

Visual hierarchy: The designed UI aims to allow the project manager as the end user to concentrate on what is more important. The used size, color, location of the items, spatial relationships between items, and the structure of the page all together try to make the project manager to understand the designed UI.

Visibility: The design makes all of the information required for completing a task visible to the project manager without extracting him/her. For example, when the project manager tries to estimate the communication distances between each pair of teams, she/he can see all of the required data about the corresponding teams as it is shown in Figure 12. As

the other examples of visibility, the data of the teams and components stored in the database is shown in the Teams Information Table and Components Information Table respectively. The project manager can review the data at any time and then decide about deleting or updating them if it is needed.

Communication Distances Information Table

ID	Name	Country	City	Physical Distance (0-10)	Cultural Characteristics (0-10)	Time-Zone Shift (0-10)	Input Summary	Overall Distance (0-10)	
FI,SUO,896#LA,RIG,604	Suomi_12A vs. Riga_D	Finland vs. Latvia	Turku vs. Riga						Save
ES,TAL,879#LA,RIG,604	Tallin_A vs. Riga_D	Estonia vs. Latvia	Tallinn vs. Riga				PD: 1 CC: 2 TZS: 0	1.00	Save
ES,TAL,879#FI,SUO,896	Tallin_A vs. Suomi_12A	Estonia vs. Finland	Tallinn vs. Turku						Save

Figure 12. Visibility of the teams' information required to estimate the communication distances

Separation of the information: The designed UI separates the information about the teams from the GSD projects. The reason for the separation of the information is that the same teams with the same information may execute several GSD projects.

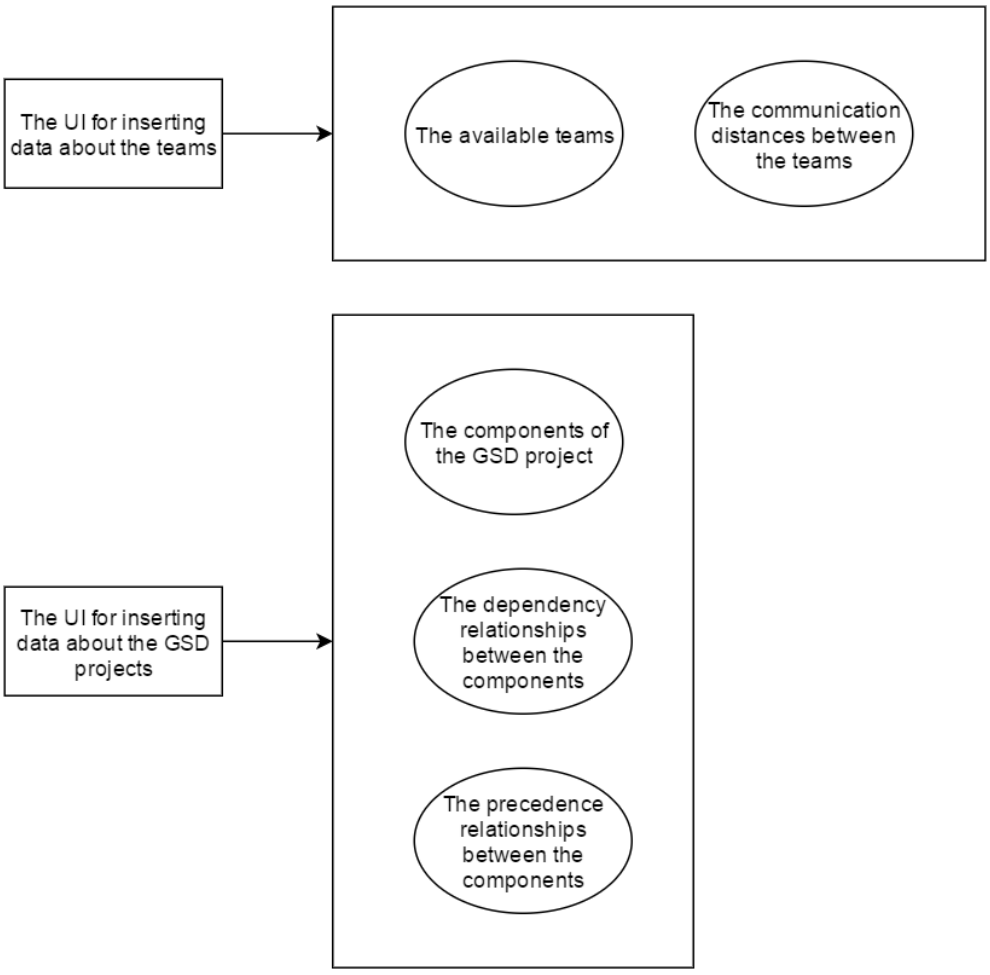


Figure 13. Separation of the information and five major parts of the developed UI

Figure 13 shows the separation of information in the developed UI. As a result, the developed UI consists of five major parts that are listed in the following.

1. The UI for introducing the GSD teams
2. The UI for estimating the communication distances between the available teams
3. The UI for introducing the GSD projects in terms of logical components
4. The UI for determining the dependency relationships between the components
5. The UI for determining the precedence relationships between the components

In the following sections, the different parts of the developed UI are explained and clarified in detail.

4.3.1 User Interface for Introducing the Available Teams

The first section of the developed application allows the project manager to gather data about the available GSD teams. As it can be seen from Figure 14, the project manager can add, delete, and update the GSD teams. Also, the UI provides a Teams Information Table to enable the project manager to observe the data of the teams stored into the database. In case of adding a new team, or deleting or updating an available team, the Teams Information Table is updated and shows the latest status of the teams stored in the database.

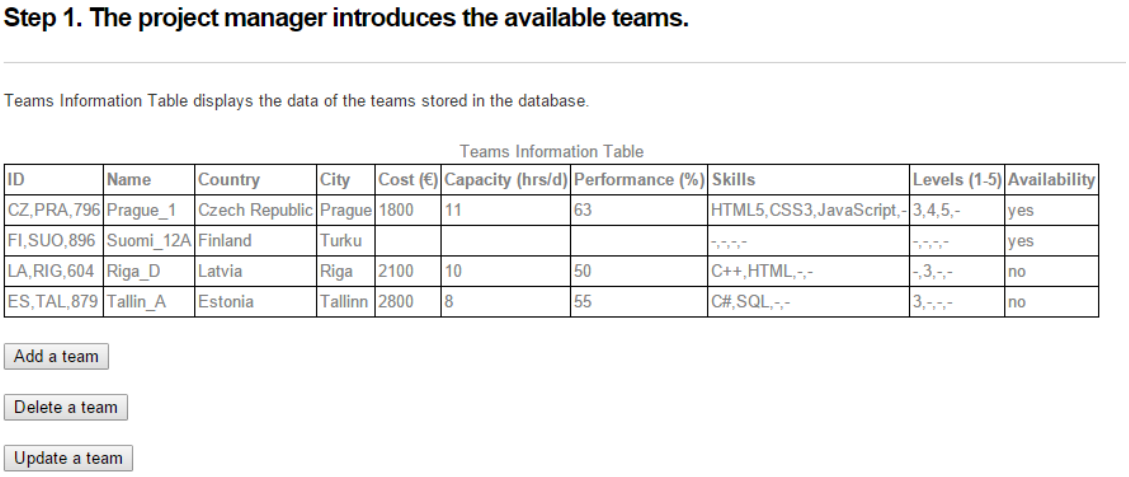


Figure 14. The UI for introducing the GSD teams

Adding a new team: In order to add a new team into the database, the project manager clicks on the "Add a team" HTML button. Next, an HTML form appears. Figure 15 shows the form containing different fields of data about an available team. After filling out the form, the project manager saves the team into the database by clicking on the "Save" button. Filling out some of the input fields of the form such as Name, Country of Site, City of Site is mandatory. Hence, a team cannot be saved into the database without filling out the mandatory input fields.

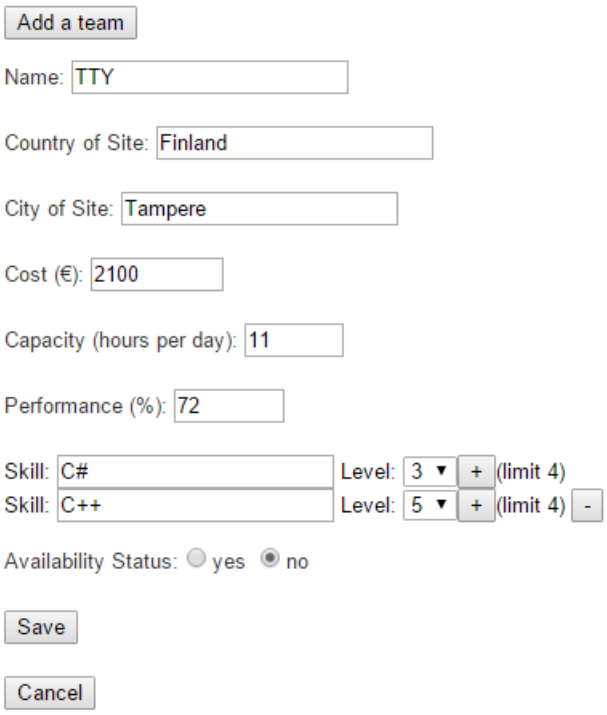


Figure 15. HTML form for adding a new team to the database

Figure 16 illustrates a JavaScript alert that appears when the project manager tries to save the team into the database without filling out the compulsory fields. After clicking on the "ok" button of the JavaScript alert, the missing fields of the form are addressed, and the project manager is reminded to fill out those fields, as it can be seen in Figure 17.

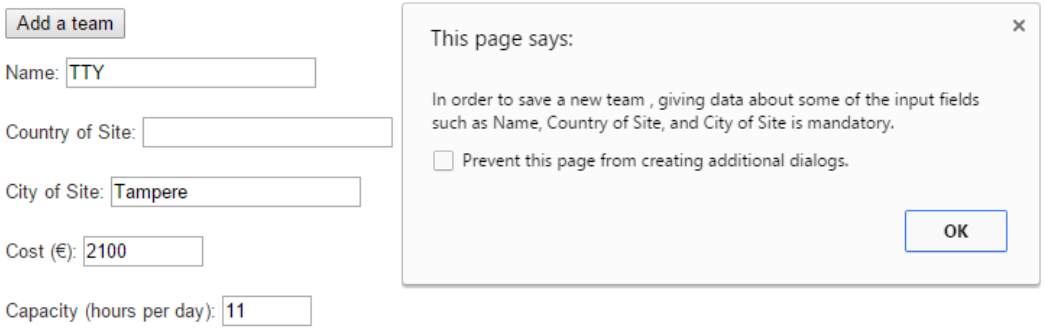


Figure 16. The JavaScript Alert for the compulsory missing input fields

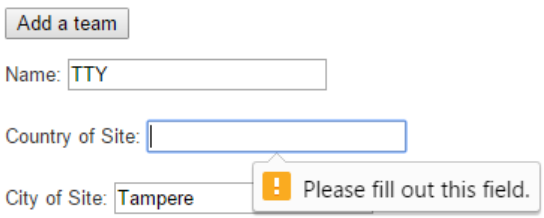


Figure 17. Reminding the project manager to fill out the compulsory input fields by addressing them

The UI provides a range of legal values for some of the input fields such as Cost, Capacity and Performance. Moreover, the UI allows the project manager to select only one of the two choices of "yes" or "no" for the input field of Availability Status. Furthermore, since the number of skills for a team is not a fixed number, the UI allows the project manager to add or remove the extra fields as it can be seen in Figure 18. Also, the project manager can specify a level for each skill in the range of 1 up to 5 from a drop-down list.

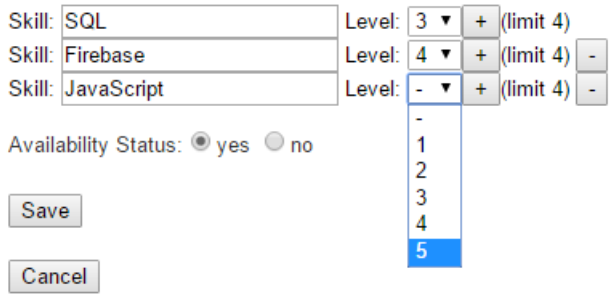


Figure 18. Adding or removing the Skill input fields

After filling out all of the mandatory input fields, the project manager clicks on the "Save" button to store the team into the database. Next, the information of the new team is shown in the Teams Information Table, as it can be seen in Figure 19. When adding a new team into the database, a unique ID (Identity) for the new team is generated automatically. This unique ID will be used by the project manager when deleting or updating a team.

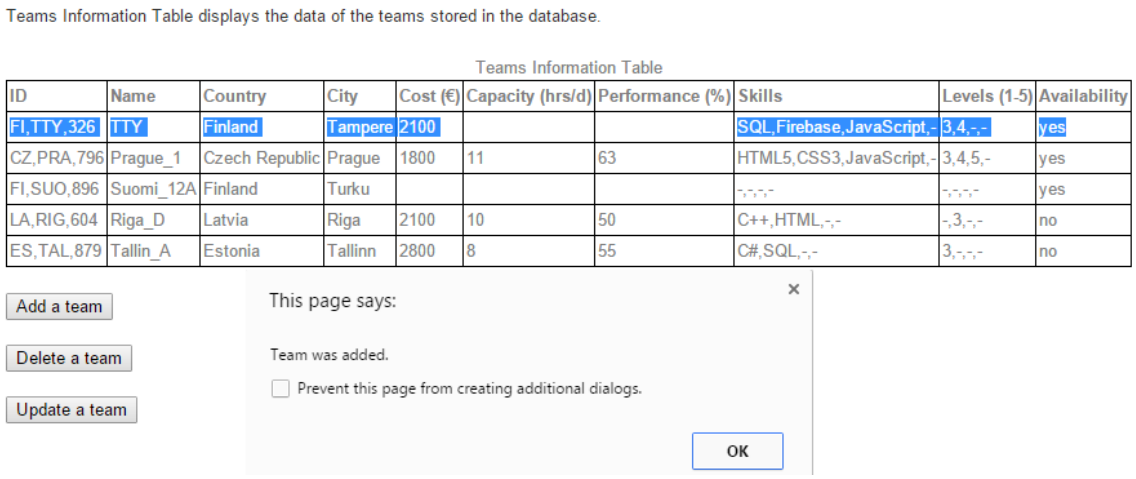
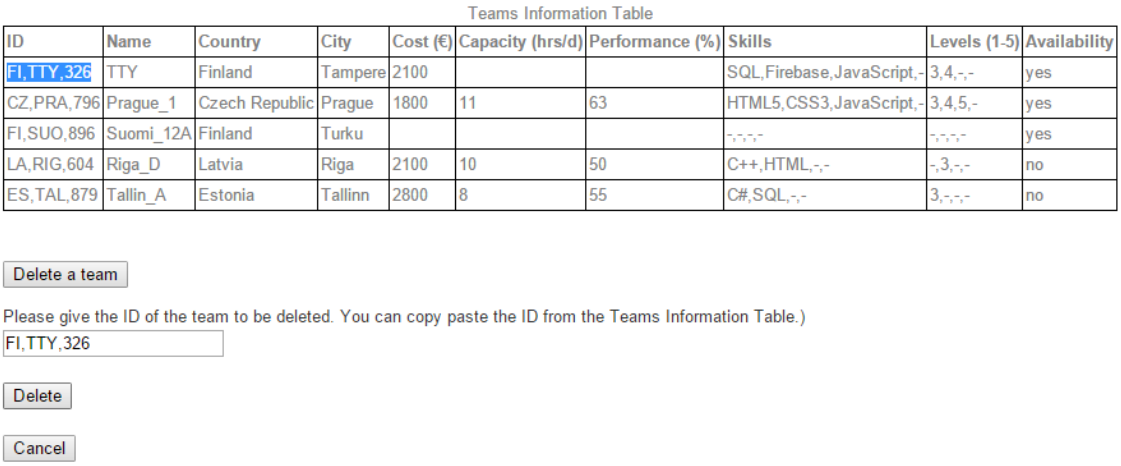


Figure 19. Displaying the new team in the Teams Information Table

Deleting a team: In order to delete a team from the database, the project manager clicks on the "Delete a team" button. Next, an input field appears and asks the project manager to enter the ID of the team to be deleted. The project manager can copy paste the corresponding ID from the Teams Information Table to the input field. Next, by clicking on the "Delete" button, the application searches in the database for the team with the given ID. In case of finding the given ID, the application deletes the corresponding team from the database and then informs the project manager by a JavaScript alert. If the given ID

is not found in the database, the project manager receives a JavaScript alert expressing that the given ID is not valid.



Teams Information Table

ID	Name	Country	City	Cost (€)	Capacity (hrs/d)	Performance (%)	Skills	Levels (1-5)	Availability
FI,TTY,326	TTY	Finland	Tampere	2100			SQL,Firebase,JavaScript,-	3,4,-,-	yes
CZ,PRA,796	Prague_1	Czech Republic	Prague	1800	11	63	HTML5,CSS3,JavaScript,-	3,4,5,-	yes
FI,SUO,896	Suomi_12A	Finland	Turku				-,-,-,-	-,-,-,-	yes
LA,RIG,604	Riga_D	Latvia	Riga	2100	10	50	C++,HTML,-,-	-,3,-,-	no
ES,TAL,879	Tallin_A	Estonia	Tallinn	2800	8	55	C#,SQL,-,-	3,-,-,-	no

Figure 20. Deleting a team from the database

Updating a team: In order to update a team in the database, the project manager clicks on the "Update a team" button. Next, an input field appears and asks the project manager to enter the ID of the team to be updated.

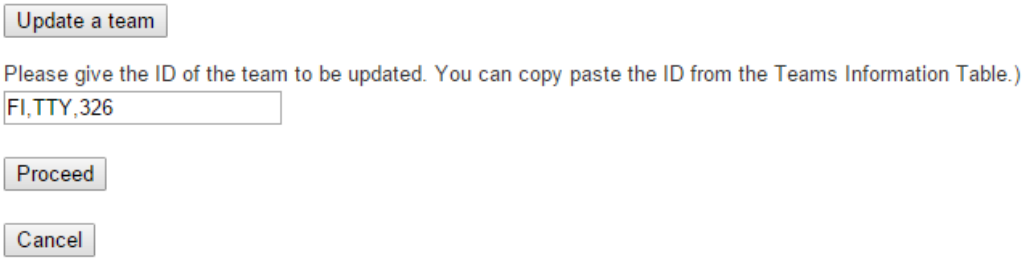


Figure 21. Applying the update operation

By clicking on the "Proceed" button, the application searches in the database for the team with the given ID. In case of not finding the corresponding ID, the project manager receives a JavaScript alert. Otherwise, the data of the team with the corresponding ID is retrieved from the database and displayed in an HTML form. Some of the input fields such as ID, Name, Country of Site, and City of Site are not allowed to be updated by the project manager. After updating the data of the team, the project manager clicks on the "Update" button to save the updated input fields into the database. Next, the project manager receives a JavaScript alert confirming the update operation. Furthermore, the data of the updated team is simultaneously refreshed in the Teams Information Table. Figure 22 represent the HTML form to be interacted with by the project manager to update a team in the database.

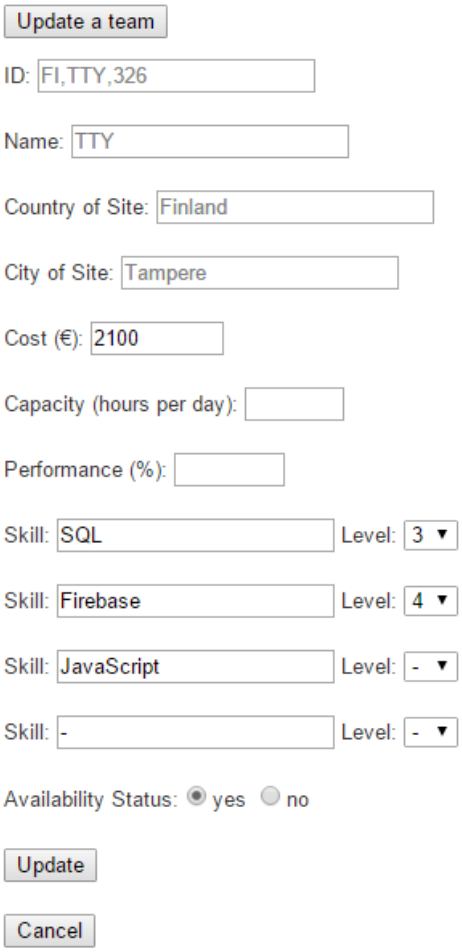


Figure 22. The HTML form for updating a team in the database

4.3.2 User Interface for Estimating the Communication Distances

A numeric value for the communication distance between each pair of teams should be estimated by the project manager based on multiple factors such as physical distance, cultural characteristics, and time-zone shift between the teams. The application provides a Communication Distances Information Table for the project manager to interact with in order to specify the values of the communication distances. Figure 23 shows the Communication Distances Information Table. The application displays all of the possible communication distances between the teams in the table by creating several data fields for each communication distance such as ID, Name, Country, and City. The project manager specifies some values for the input fields of Physical Distance, Cultural Characteristics, and Time-Zone Shift by interacting with the slider control items in the table. After clicking on the "Save" button, a summary of the given values as well as an overall communication distance (the average of the distances) are shown in the table, as it can be seen in Figure 24. The project manager can update the input fields of a communication distance at any time by interacting again with the slider control items and then clicking on the "Save" button.

Step 2. The project manager estimates the communication distances between the available teams.

The project manager interacts with the Communication Distances Information Table in order to estimate the communication distances between the teams. Moreover, the table displays the data of the communication distances stored in the database.

Communication Distances Information Table

ID	Name	Country	City	Physical Distance (0-10)	Cultural Characteristics (0-10)	Time-Zone Shift (0-10)	Input Summary	Overall Distance (0-10)	
FI,TTY,326#LA,RIG,604	TTY vs. Riga_D	Finland vs. Latvia	Tampere vs. Riga						Save
FI,SUO,896#LA,RIG,604	Suomi_12A vs. Riga_D	Finland vs. Latvia	Turku vs. Riga						Save
FI,SUO,896#FI,TTY,326	Suomi_12A vs. TTY	Finland vs. Finland	Turku vs. Tampere						Save
ES,TAL,879#LA,RIG,604	Tallin_A vs. Riga_D	Estonia vs. Latvia	Tallinn vs. Riga				PD: 1 CC: 2 TZS: 0	1.00	Save
ES,TAL,879#FI,TTY,326	Tallin_A vs. TTY	Estonia vs. Finland	Tallinn vs. Tampere						Save
ES,TAL,879#FI,SUO,896	Tallin_A vs. Suomi_12A	Estonia vs. Finland	Tallinn vs. Turku						Save
CZ,PRA,796#LA,RIG,604	Prague_1 vs. Riga_D	Czech Republic vs. Latvia	Prague vs. Riga						Save
CZ,PRA,796#FI,TTY,326	Prague_1 vs. TTY	Czech Republic vs. Finland	Prague vs. Tampere						Save
CZ,PRA,796#FI,SUO,896	Prague_1 vs. Suomi_12A	Czech Republic vs. Finland	Prague vs. Turku						Save
CZ,PRA,796#ES,TAL,879	Prague_1 vs. Tallin_A	Czech Republic vs. Estonia	Prague vs. Tallinn						Save

Figure 23. The Communication Distances Information Table

Communication Distances Information Table

ID	Name	Country	City	Physical Distance (0-10)	Cultural Characteristics (0-10)	Time-Zone Shift (0-10)	Input Summary	Overall Distance (0-10)	
FI,TTY,326#LA,RIG,604	TTY vs. Riga_D	Finland vs. Latvia	Tampere vs. Riga				PD: 3 CC: 4 TZS: 0	2.33	Save

Figure 24. Saving the communication distance

4.3.3 User Interface for Introducing the Logical Components

As it was mentioned earlier in 3.3.1, the GSD work distribution model characterizes the system to be developed is terms of logical components. A logical component is a work package that consists of a set of responsibilities. The developed application provides a UI that allows the project manager to add, delete, and update the components in the database. Moreover, a Components Information Table is provided to display the recent status of the components stored in the database. In case of adding, deleting, or updating a component, the Components Information Table is refreshed simultaneously. Figure 25 shows the developed UI for introducing the components of the GSD project.

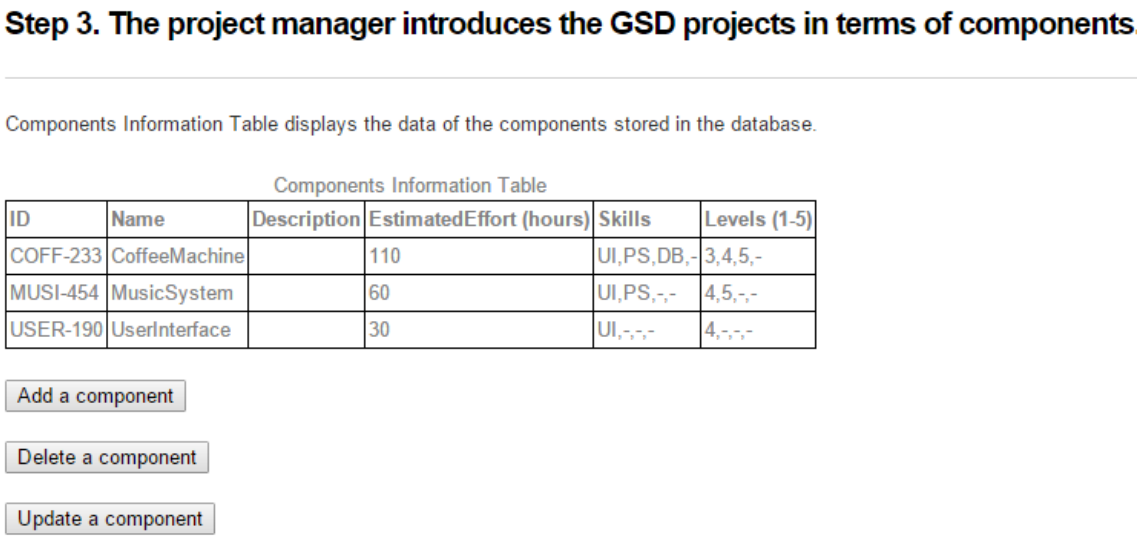


Figure 25. The UI for introducing the components of the GSD project

Adding a new component: In order to add a new component into the database, the project manager clicks on the "Add a component" button. Next, the following HTML form appears.

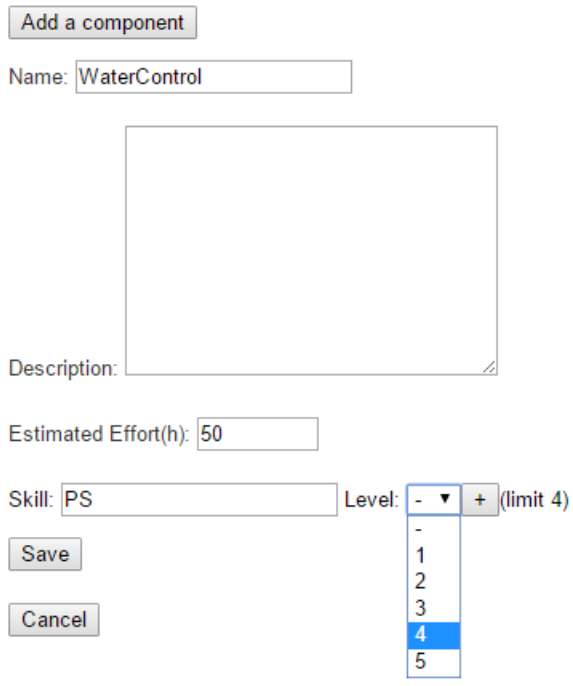


Figure 26. The HTML form for adding a new component

Filing out the input field of Name is mandatory, otherwise the component cannot be saved into the database, and the project manager is informed by receiving a JavaScript alert. After filling out the input fields, the project manager clicks on the "Save" button to store the component into the database. Next, the stored component is shown in the Components Information Table. Also, the application generates a unique ID automatically for the

stored component. This unique ID will be used by the project manager when deleting or updating the component.

Deleting a component: In order to delete a component from the database, the project manager clicks on the "Delete a component" button. Next, the following HTML form appears and asks the project manager to enter the ID of the component to be deleted.

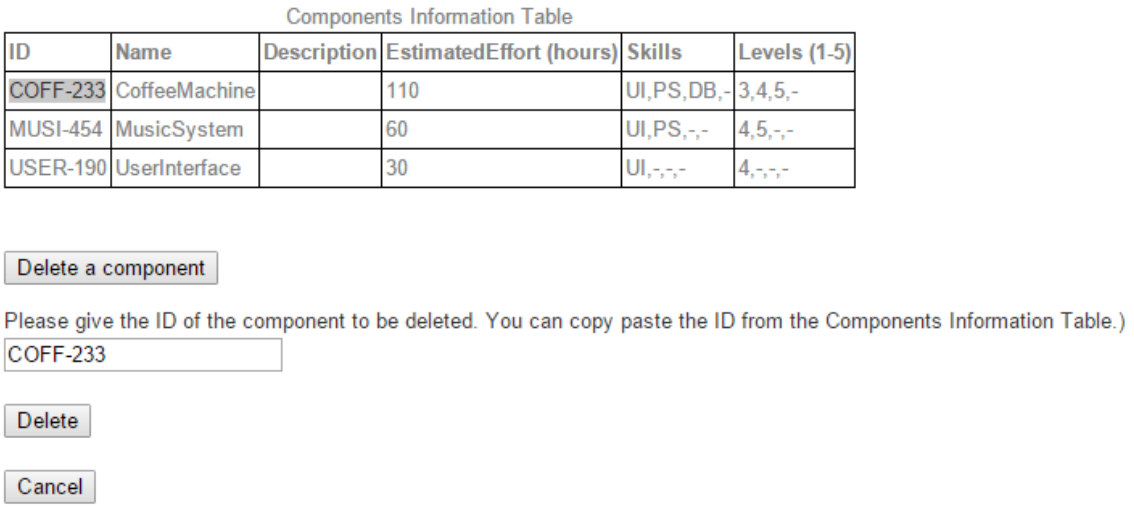


Figure 27. Deleting a component from the database

Updating a component: In order to update a component, the project manager clicks on the "Update a component" button. Next, the following form appears and asks the project manager to enter the ID of the component to be updated.

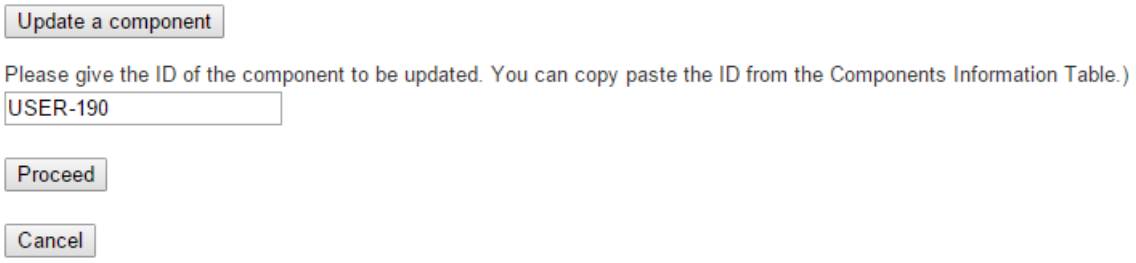


Figure 28. Applying the update operation

After clicking on the "Proceed" button, the data of the component with the given ID is retrieved from the database and appears in an HTML form, as it can be seen in Figure 29. The project manager is not allowed to update some of the input fields such as ID and Name. After filling out the input fields, the project manager clicks on the "Update" button to update the component in the database. The data of the updated component is refreshed simultaneously in the Components Information Table.

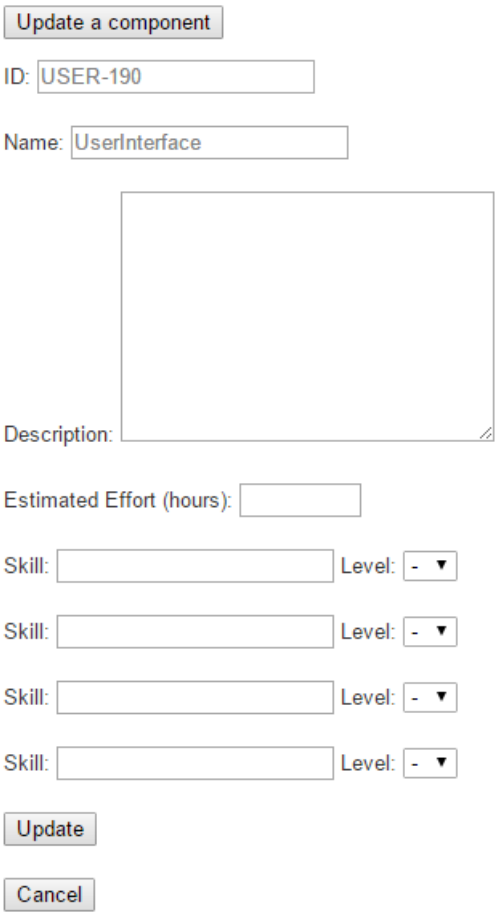


Figure 29. The HTML form for updating a component

4.3.4 User Interface for Specifying the Relationships between the Components

The developed application uses the HTML checkbox input fields in order to allow the project manager to specify the relationships (dependency and precedence) between the components of the GSD project, as it can be seen in Figure 30. After checking the components that have a relationship with a specific component, the project manager clicks on the "Save" button to store the specified relationships into the database.

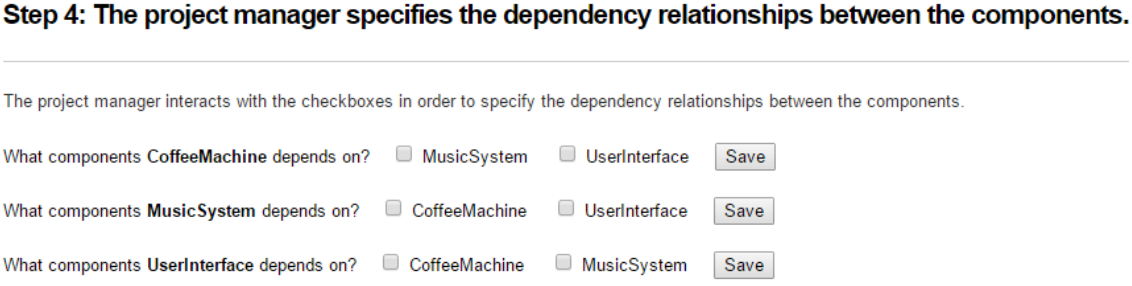


Figure 30. The UI for specifying the dependency relationships between the components

5. DEVELOPMENT

5.1 Methodology

Development of the application was based on iterative and incremental type of development. The basic idea of applying this type of development was to develop the application in smaller contributions at a time through several iterations. In each iteration, the development of the application got benefits from learnings gained during the earlier iterations. The development of the application was started by an initial implementation of a subset of major functional requirements. Next, the application was enhanced iteratively until the full application was developed.

In more detail, the development of the application was done in four iterations. The duration of each iteration was between two and three weeks, except the first iteration that took one month. At the end of each iteration, a meeting with the research group was held to get feedback from them. A web-based project management application called Trello [35] was used to manage the development process of the project. For each iteration of the development process, a Trello board was created. Each Trello board contains several lists such as to-do list, doing list, and done list. Cards inside the lists were moved gradually from one list to another (for example from to-do list to doing list). Figure 31 shows a Trello board related to the 4th iteration of the development process.

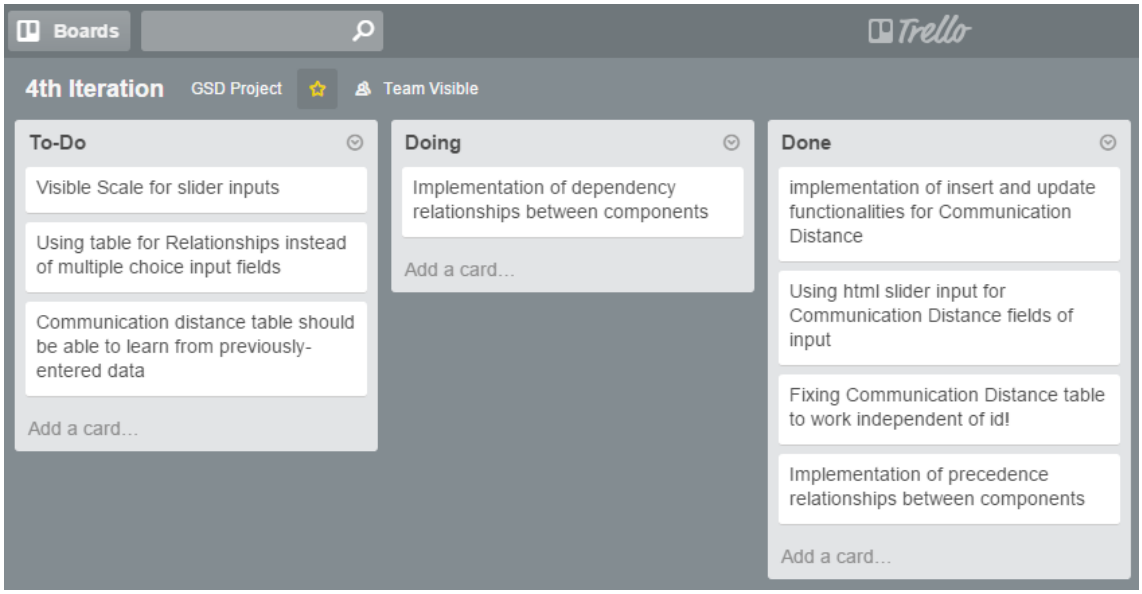


Figure 31. The Trello board for the 4th iteration of the development process

5.2 Programming Languages

HTML5, CSS3, and JavaScript were utilized as the programming languages of the developed application.

- HTML (HyperText Markup Language) [3] defines the basic structure of the developed application.
- CSS (Cascading Style Sheets) [4] takes care of the visual styling and presentation of the content of the developed application.
- JavaScript [5] as a scripting client-side language adds some dynamic elements to the developed web application that runs on the client-side.

Zen coding [36] is strongly used during the development of the HTML code. Zen Coding is an editor plugin for high-speed HTML, XML (Extensible Markup Language), and XSL (Extensible Stylesheet Language) coding and editing. Zen coding allows to make the HTML code rapidly. As an example, Program 1 in the following is a line of Zen code that develops the HTML code represented in Program 2.

```
.wrap>header>h1{My Website}+nav>ul>li*4>a[href=#]
```

Program 1. An example of Zen coding

```
<div class="wrap">
      <header>
            <h1>My Website</h1>
            <nav>
                  <ul>
                        <li><a href="#"></a></li>
                        <li><a href="#"></a></li>
                        <li><a href="#"></a></li>
                        <li><a href="#"></a></li>
                  </ul>
            </nav>
      </header>
</div>
```

Program 2. The HTML code developed rapidly by utilizing Zen coding

5.3 Database

The developed web application required to utilize a suitable database in order to organize and collect the data about the GSD project and the available GSD teams. Several alternatives such as Firebase database [6], HTML5 local Storage [37], and HTML5 IndexedDB [38] were briefly studied to make a decision on selecting the most suitable one for the developed web application.

Firebase [6] is a platform for mobile and web application used to store data to the cloud. Firebase follows the Backend as a Service (BaaS) model. The BaaS model provides web and mobile developers with the possibility of establishing a link between the application and the hosted backend storage. BaaS model focuses on decreasing the complexity related to the cloud as well as providing a simple API (Application Programming Interface). Several motivations promoting the usage of BaaS systems are listed in the following. [39]

- Writing and hosting a cloud backend is a difficult and expensive task.
- Some developers do not have the required skills for writing and hosting their own cloud backend.
- Working with the major backend providers such as Amazon might be complex for some unskilled developers.
- Working with the BaaS systems is almost simple.

The above mentioned motivations mostly point to the advantages of using Firebase as a BaaS system instead of writing a separate backend to manage the database. So, they do not necessarily compare the Firebase to HTML5 local storage or IndexedDB.

With HTML5 local storage, it is possible to store data locally in the user's browser. There are some advantages and disadvantages. Local storage is secure. Also, it allows to store large amount of data without decreasing the performance of the website. However, local storage on large datasets may result in poor performance. With local storage, data is unstructured with no facilities for searching and indexing. [37] [40]

HTML5 IndexedDB provides a transactional high-performance data store. Also, it provides local data persistency which allows the web applications to be accessible both offline and online. It is suitable for robust client-side data storage and access. It supports modern desktop browsers. However, it does not support completely the old browsers. Its API is new, large, and complex. So, working with IndexedDB would be slightly difficult. [38] [40]

As a conclusion to this section, after carefully considering the advantages and disadvantages the above mentioned alternatives, the Firebase as a BaaS service was chosen to handle the data of the developed web application.

5.3.1 Firebase and its Features

The developed application utilizes a Firebase database to store, retrieve, update, and synchronize the data about the GSD project and the participating teams. Some of the important features of Firebase are listed in the following. [41]

- Firebase provides real-time databases. A real-time database facilitates the real-time transactions by considering explicit time constraints such as deadlines [42].

- Data in the Firebase database is stored in JSON documents, and synchronized in real-time to the connected clients. JSON [43] [44] stands for JavaScript Object Notation and that is a light-weight data interchange format. Some of the advantages of the JSON format are listed in the following.
    1. JSON is language-independent. However, it is based on a subset of the JavaScript Programming Language. Hence, JSON is very convenient for data interchange in JavaScript applications.
    2. JSON is self-describing.
    3. JSON is easy to understand.
- Firebase ensures the security of data. Database access and validation are secure. Data transfers over a secure SSL connection.
- Firebase works offline. Client applications using the Firebase stay responsive regardless of latency of network or internet connectivity. Writing to a Firebase database results in triggering the local events directly, before the data is written to the server. Once connectivity is maintained again, the client receives any missing changes. So, the client is synchronized with the current state of the server.
- Firebase provides user authentication. There is a built-in functionality for authenticating users in different ways such as email and password, Facebook, Twitter, GitHub, Google, and anonymous authentication.

Firebase has a comprehensive archive of documents for developers regarding all different services it provides. These services can be seen in Figure 32. Firebase supports a web platform by providing a JavaScript API.

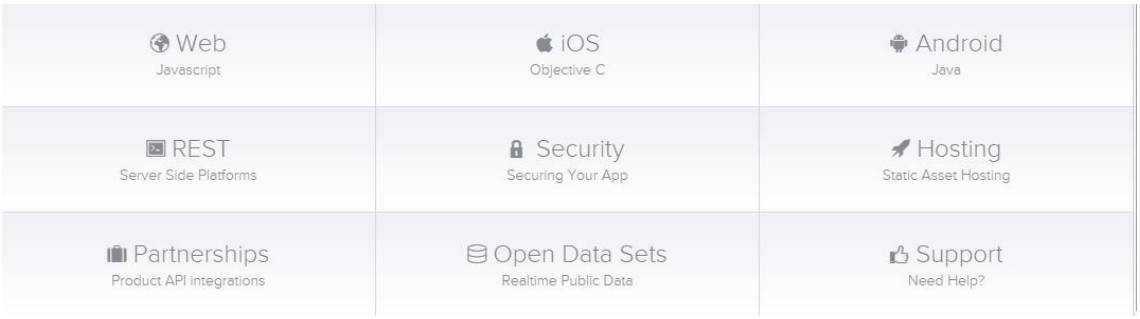


Figure 32. Firebase Developer Documents

5.3.2 Basics of How Firebase Works

Signing up: After signing up for a free account, a free Firebase application is created which has a unique database URL (Uniform Resource Locator) ending in `firebaseio.com`. This unique URL is used to store the data.

Installation: In order to include the Firebase JavaScript client library in the web application directly from Firebase CDN (Content Delivery Network), following script tag needs to be added to the `<head>` section of the HTML file [45].

```
<script src="https://cdn.firebase.com/js/client/2.3.1/firebase.js"> </script>
```

Program 3. Firebase JavaScript client library installation

Reference to the database: in order to read or write data, a reference to the database (in other words a path to database) should be created. A Firebase reference represents a particular location in the database used for reading or writing data to that location. However, creating a reference does not create a connection to the server or begin downloading data. The primary Firebase reference is created from a full and secure Firebase database URL by calling new Firebase(firebaseURL). However, the user can subsequently reference child locations by calling child(childPath) on an existing Firebase reference. [46]

Firebase Methods: Table 2 represents some of the most important Firebase methods used to reference a location in the Firebase database. [47]

Table 2. Firebase Methods for referencing a location

Firebase Method	**Functionality**	**Arguments**
new Firebase(firebaseURL)	Creating a new Firebase reference from a full Firebase URL	HTTPS URL of a Firebase location
child(childPath)	Getting a Firebase reference to a location at the specified relative path	A relative path from this location to the desired child location
parent( )	Getting a Firebase reference to the parent location	
root( )	Getting a Firebase reference to the root of the Firebase	
key( )	Returning the last token in a Firebase location.	
name( )	Returning the last token in a Firebase location	

Understanding the data: As it was mentioned earlier in 5.3.1, data in the Firebase database is stored in a JSON document. Figure 33 represents an example of a JSON tree in the Firebase database of the developed web application, and Program 4 represents the corresponding code. Firebase allows the users to import and export data in a JSON document [46].

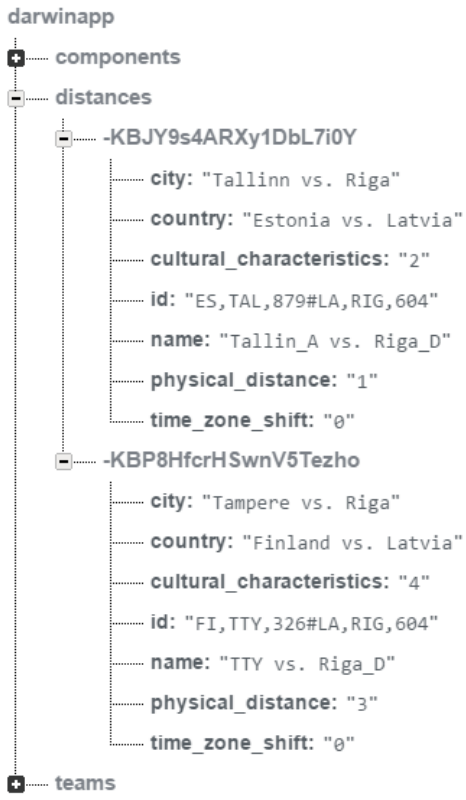


Figure 33. Storing data as a JSON tree in the Firebase database

```
"distances" : {
 "-KBJY9s4ARXy1DbL7i0Y" : {
   "city" : "Tallinn vs. Riga",
   "country" : "Estonia vs. Latvia",
   "cultural_characteristics" : "2",
   "id" : "ES,TAL,879#LA,RIG,604",
   "name" : "Tallin_A vs. Riga_D",
   "physical_distance" : "1",
   "time_zone_shift" : "0"
 },
 "-KBP8HfcrHSwnV5Tezho" : {
   "city" : "Tampere vs. Riga",
   "country" : "Finland vs. Latvia",
   "cultural_characteristics" : "4",
   "id" : "FI,TTY,326#LA,RIG,604",
   "name" : "TTY vs. Riga_D",
   "physical_distance" : "3",
   "time_zone_shift" : "0"
 }
}
```

Program 4. The corresponding JavaScript code for the JSON tree represented in Figure 33

5.3.3 Saving Data to the Firebase Database

The developed application needs to save data into the firebase database in several different circumstances. Some of them are listed in the following.

- When the project manager adds or updates the data of a team/component
- When the project manager determines or updates the values of communication distances
- When the project manager specifies or updates the relationships (dependency and precedence) between the components of the GSD project

Firebase provides four methods to save data in the database. Table 3 represents the methods and their functionalities. [48]

1. The basic operation for writing to the Firebase database is set( ) that stores new data to a specified database reference (database location). Using set( ) will overwrite any existing data on the specified database location. Program 5 shows creating a reference to a database location. Also, Program 6 shows a write operation to a child node of the created reference.

```
var ref = new Firebase("https://docs-examples.firebaseio.com/
web/saving-data/fireblog");
```

Program 5. Creating a reference to the database location

```
var usersRef = ref.child("users");
usersRef.set({
  arvinj: {
    date_of_birth: "June 17, 1984",
    full_name: "Arvin Jalali"
  },
  farnazph: {
    date_of_birth: "June 16, 1974",
    full_name: "Farnaz Pashmforoosh"
  }
});
```

Program 6. Using set( ) to write to a child node

2. In order to write to the several children of a database location simultaneously without facing some issues due to overwriting the other child nodes, update( ) should be used instead of set( ). Program 7 shows updating the user farnazph by using update( ). In case of using set( ) instead of update( ), then some of the data fields of the user farnazph such as full_name and date_of_birth will be deleted.

```
var myRef = usersRef.child("farnazph");
myRef.update({
  "nickname": "Feri"
});
```

Program 7. Using update( ) to update a child node without overwriting data

3. Considering the multi-user nature of many applications, when creating a list of data by multiple users at a same time by using set( ), there can be the possibility of write conflicts. Hence, the Firebase provides a push( ) method to generate a unique ID (or key) for each new child. By using the unique child keys, multiple clients are allowed to add children to the same location at a same time without being worried about the write conflict issues.
4. Firebase provides a transaction( ) method in order to avoid from the corruption of complex data due to some concurrent modifications.

Table 3. Firebase Methods for saving data to the database

Method	**Functionality**
set( )	Writing or replacing the data to a specified path `messages/users/<username>`
update( )	Updating some of the keys for a specified path without replacing the whole of data
push()	Adding a list of data, and generating a unique ID for each item `messages/users/<unique-user-id>/<username>`
transaction( )	Preventing the complex data from being corrupted due to some concurrent updates

5.3.4 Retrieving Data from the Firebase Database

The developed application needs to retrieve data form the Firebase database for several different purposes, as they are listed in the following.

- The data of the teams stored in the Firebase database is retrieved and presented in the Teams Information Table.

- The data of the components stored in the Firebase database is retrieved and presented in the Components Information Table.
- The determined values of the communication distances stored in the Firebase database are retrieved and presented in the Communication Distances Information Table.
- The specified relationships between the components of the GSD project stored in the Firebase database are retrieved and presented by the application.
- When the project manager tries to update a team, the data of that team is retrieved from the Firebase database and presented in the team's update HTML form.
- When the project manager tries to update a component, the data of that component is retrieved from the Firebase database and presented in the component's update HTML form.

Firebase retrieves the stored data by attaching an asynchronous listener to a database reference. The attached listener is triggered once for the initial state of the data, and again when the data changes. This section presents the basics of retrieving data from a Firebase database by explaining four different types of read event such as value event, child added event, child changed event, and child removed event. [49]

1. Value Event: The value( ) event reads a static snapshot of the contents at a specified location of the database. The value event is triggered once to read the initial data, and then it will be triggered again whenever data changes. So, in case of adding new data to the database reference, there is no need to write some extra code to read that. The callback function of the value event returns a DataSnapshot which is a snapshot of the data. A snapshot is a picture of the data at a particular database location at a particular time. By calling val( ) on a snapshot, the JavaScript object representation of the data is returned. The value of the snapshots is null when there is no data at that location. Program 8 shows an example of using the value event to receive a snapshot of the data stored in the Firebase database. Also, Table 4 represents some of the useful DataSnapshot methods of Firebase.

```
// Get a database reference to our posts
var ref = new Firebase("https://docs-examples.firebaseio.com/web/saving-
data/fireblog/posts");

// Attach an asynchronous callback to read the data at our posts reference
ref.on("value", function(snapshot) {
  console.log(snapshot.val());
}, function (errorObject) {
  console.log("The read failed: " + errorObject.code);
});
```

Program 8. Example of value event to read data from a Firebase database [49]

2. Child Added Event: By using the child_added( ), it is possible to retrieve a list of items from a Firebase database. The child_added( ) is triggered once for each existing child and then again when a new child is added to that location, while the value( ) returns the entire contents of the specified location. The child_added( ) callback returns a snapshot containing the data of a new child.
3. Child Changed Event: The child_changed( ) is triggered in case of modifying a child node including any modifications to descendants of the child node.
4. Child Removed Event: The child_removed( ) is triggered in case of removing a child.

Firebase database events always guarantee the features that are listed in the following. [49]

- When local state changes, events will always be triggered.
- Events always retrieve the correct state of the data. However, there can be some cases in which local operations or timing results in temporary differences. For example, temporary loss of network connectivity can result in such cases.
- Writes from a client are always be committed to the server, and then distributed to the other users.
- Value events always contain updates from any other events which have been happened before that value event.

Table 4. DataSnapshot Methods

DataSnapshot Method	**Functionality**	**Arguments**
exists( )	Returns true if this DataSnapshot contains any data.	
val( )	Gets the JavaScript object representation of the DataSnapshot.	
child(childPath)	Gets a DataSnapshot for the location at the specified relative path.	A relative path to the location of child data
hasChild(childPath)	Returns true if the specified child exists.	A relative path to the location of a potential child

hasChildren( )	Returns true if the DataSnapshot has any children.	
key( )	Gets the key of the location that generated this DataSnapshot.	
ref()	Gets the Firebase reference for the location that generated this DataSnapshot.	

Queries: by using Firebase database queries, it is possible to retrieve data selectively based on different factors. In order to make a query in the Firebase database, the user should specify how he/she prefer the data to be ordered using one of the methods represented in Table 5. [49]

Table 5. Firebase query Methods

Query Method	**Functionality**	**Arguments**
on(eventType, callback)	Listens for data changes at a particular location.	Event type string, callback function to pass a DataSnapshot
once(eventType, successCallback)	Listens for exactly one event of the specified event type, and then stops listening.	Event type string, callback function to pass a DataSnapshot
orderByChild(key)	Generates a new Query object ordered by the specified child key.	The child key to order by.
orderByKey( )	Generates a new Query object ordered by key.	
orderByValue( )	Generates a new Query object ordered by child values.	
orderByPriority( )	Generates a new Query object ordered by priority.	

5.4 Software Testing

Currently, there is no link between the developed UI (web application) and the SB-GSD tool as the actual tool, since that is not in the scope of the thesis. Furthermore, the developed UI is a preliminary small-scale prototype of the SB-GSD tool, and the UI should be modified significantly before linking to the actual tool. Hence, the developed UI has not necessarily required any strict testing procedure so far. As a result, the exploratory manual testing was practiced to test the developed UI.

As it was mentioned earlier in 4.3, the developed UI consists of five major parts. During the development process, each major part of the UI was tested individually as well as each time they were integrated. Testing was done by giving valid data input to explore whether the output is correct or not, as well as by giving invalid data input to explore the unexpected conditions.

At the end of the development, the web application was transferred over the internet by using a free web hosting. Next, by performing an informal self-evaluation, the developer tried to play the role of the project manager (as the end user) to examine how well all of the features of the application ensure the expected behaviors. In summary, the self-evaluation reflects that the developed application fulfills all of the main features of the UI claimed in 4.1. Also, the UI is simple, and easy to understand for the project manager. However, the UI can be improved significantly to provide a better user experience for the project manager. In the next chapter, the evaluation of the developed UI will be discussed in detail.

6. EVALUATION OF THE DEVELOPED UI

This chapter evaluates the UI (web application) developed in the thesis. At a same time, some further works are proposed to improve the application. Firstly, the web application was transferred over the internet by utilizing a free web hosting. Next, a self-evaluation was done. The self-evaluation of the developed UI shows that all of the main features declared in 4.1 were achieved. The UI allows the project manager to introduce the GSD projects (in terms of components) as well as the participating GSD teams. Also, the project manager can estimate the communication distances between the teams and determine the dependency and precedence relationships between the components of the GSD projects.

The UI seems to be simple and easy to understand particularly for the project manager as the end user. The UI provides consistency in the used language, elements, and layouts. The UI attempts to provide a visual hierarchy for the project manager by choosing suitable size, color, and location of the items. The spatial relationships between the items and the structure of the page try to make the project manager to understand the UI. However, there can be some significant improvements in the special relationships between the items. For example, by adding more new teams, the number of communication distances between teams grows dramatically. Hence, the number of rows in the Communication Distances Information Table grows considerably. So, there can be an imbalance in the special relationships between the items. The UI makes all of the required information for doing the tasks visible to the project manager. The UI separates the information about teams from the information about the GSD projects. So, the same teams with the same information can execute multiple GSD projects.

The developed UI can be improved to provide a better user experience for the project manager. In the following some of the issues as well as the proposed improvements are discussed.

Unnecessary access to the ID of teams and components: As it was mentioned earlier in 4.3.1 and 4.3.3, when the project manager tries to delete/update teams and components, she/he needs to have access to the unique IDs of the items which are generated automatically when adding new teams and components. Figure 34 shows the required ID to delete a component from the database. In a good user experience, the project manager should not deal with unnecessary information such as the IDs of teams and components. This issue is referenced to the design and implementation of the Teams Information Table and Components Information Table which were done during the 1st and 2nd iterations. These two tables are not interactive tables. Hence, the project manager cannot interact with them to add, delete, and update items. However, the project manager interacts with the HTML forms to add, delete, and update items, and then the change is displayed in the content of

the tables. There are two alternative solutions proposed as further work to avoid the project manager from having access to the IDs of teams and components.

1. The Teams Information Table and Components Information Table can be designed and implemented as interactive tables. Then, the project manager can add, delete and update the items directly from the tables instead of interacting with the HTML forms. By learning from the 1st and 2nd iterations, the Communication Distances Information Table designed and implemented as an interactive table during the 3rd and 4th iterations. It can be interacted by the project manager in order to both estimate and update the distances between the teams without dealing with some unnecessary information such as the IDs of communication distances.
2. The HTML forms for deleting/updating the teams and components can be modified in a way that allows the project manager to select the ID (or Name) of the team/component to be deleted/updated from a drop down list. This can be done by retrieving the IDs of teams/components stored in the database and then displaying the IDs in a drop down input field in order to be selected by the project manager.

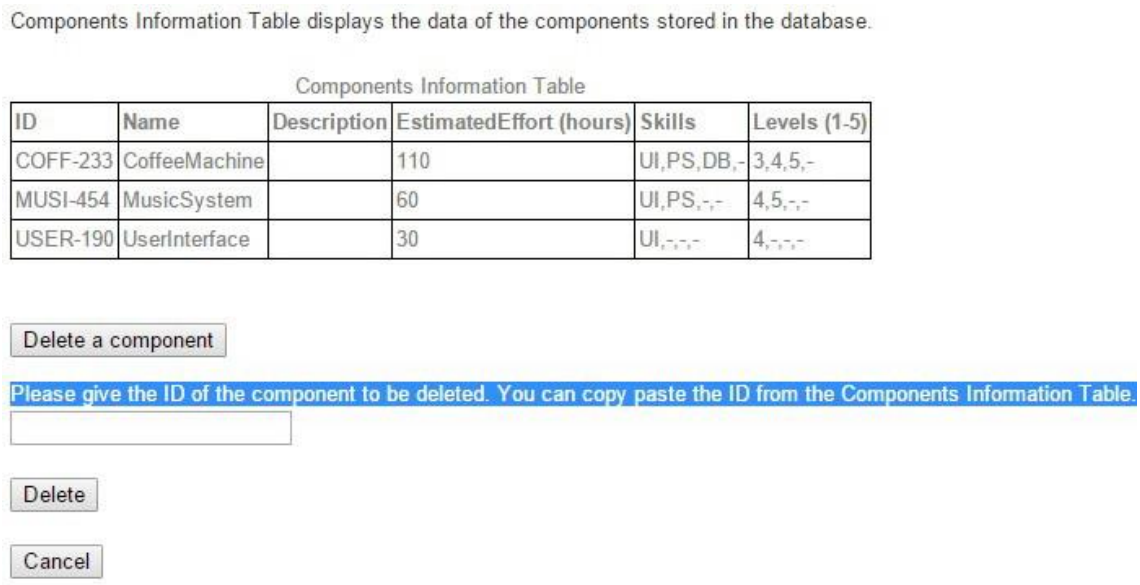


Figure 34. The project manager deals with few unnecessary information

Lack of handy sliders in the Communication Distances Information Table: As it was mentioned earlier in 4.3.2, the project manager interacts with the sliders to choose a specific value (between 0 and 10) for the different fields of distance between the teams. When, the project manager drags the slider, the numerical values corresponding to the current position of the slider cannot be observed. So, the project manager cannot have any idea about the selected specific value before saving that value, as this issue is shown in Figure 35. Hence, some further implementation (HTML/JavaScript) can improve the sliders to be more informative.

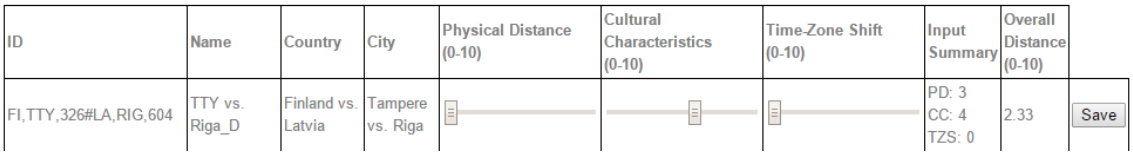


Figure 35. Lack of handy sliders

7. CONCLUSION

This master thesis generally consists of two major parts that are listed in the following.

1. Conducting a research on search-based planning of a global software development (GSD) project.
2. Prototyping a tool support (proposed by Vathsavayi) that aims to assist the project manager in planning a GSD project.

In the following, a summary table is presented. Furthermore, in the next sections, the two major parts of the thesis (research set-up and prototyping) are summarized. Finally, the lessons learned during the thesis are presented.

Table 6. Summery Table

Topic of the Thesis	Interactive Planning Tool For Global Software Projects
Background Areas	global software development, software planning, search-based software engineering
Literature Review and Research	planning GSD projects, search-based GSD project planning, concept of distance in the context of GSD
Base of the Thesis	the research of Vathsavayi including a GSD model and some ideas for developing the SB-GSD tool that assists the project manager in planning a GSD project
Prototyping the SB-GSD Tool	designing and implementing a web application as a UI for the SB-GSD tool used by the project manager when planning the GSD project
The Developed Application	a simple UI that allows the project manager to gather data about the GSD project and the participating teams

Database	The data is saved in a Firebase database. The data is added, deleted, retrieved, and updated by the project manager at any time. The stored data will be used as the inputs to the SB-GSD tool.
Programming Languages	HTML, CSS, and JavaScript
Evaluation	Evaluation of the developed UI and proposing some further works to provide a better user experience

7.1 Research Set-Up

When planning a GSD project, the project manager usually faces some problems such as how to select the teams that are more suitable for the GSD project, or how to assign work optimally to the selected teams considering the cost and duration of the project. Vathsavayi in his research proposes a GSD model and some preliminary ideas for developing an automated tool support (SB-GSD tool) that assists the project manager in planning a GSD project. Vathsavayi uses multi-objective genetic algorithms to apply search-based software engineering (SBSE) to develop the SB-GSD tool. The SB-GSD tool explores the optimal work allocations in a tradeoff between the cost and duration of the GSD project. In the thesis, it is also discussed how the GSD model and the SB-GSD tool address some of the typical issues of a GSD project such as cultural and communication issues, and task assignment issues. Furthermore, few recommendations for efficiently estimating the communication distances between the teams are proposed.

7.2 Prototyping

In this thesis, a user interface (UI) for the SB-GSD tool was designed and implemented. The developed UI is a simple, useful, and user-friendly web application used by the project manager to gather data about the GSD project and the participating teams. The data is stored into a Firebase database. The data is added, deleted, and updated by the project manager at any time as new information is provided. Consequently, the data stored in the database will be used as the inputs to the SB-GSD tool. The developed UI was finally evaluated and some improvements as further works were proposed to provide a better user experience.

7.3 Lessons Learned

Lessons learned during the thesis can be categorized in two groups. First group includes lessons learned by conducting the research on search-based planning of GSD projects focusing on the SB-GSD tool. The second group includes lessons learned by prototyping the SB-GSD tool.

Lessons learned by conducting the research on search-based planning of GSD projects are listed in the following.

- Being familiar to typical challenges of planning a GSD project
- A new insight to search-based planning of a GSD project
- How an automated tool support can be efficiently useful in planning a GSD project
- A comprehensive understanding of the concept of communication distance between teams participating in a GSD project
- Several recommendations for efficiently estimating communication distances between teams
- Addressing the GSD work distribution model besides the SB-GSD tool to some of the typical issues of a GSD project such as cultural and communication issues as well as task assignment issues

Lessons learned by prototyping the SB-GSD tool are listed in the following.

- Fulfilling a web development project in iterative and incremental model of development by learning from the feedback provided by the research group after each iteration
- Designing and implementing a concrete user interface that can be used by the project manager when planning the GSD project
- Being familiar with Firebase cloud service and working with the JavaScript API of the Firebase in order to handle the data of the developed UI
- Evaluating the developed UI and proposing some further work to provide a better user experience for the project manager

REFERENCES

[1] S. Vathsavayi, O. S. Korte and K. Systä, "Tool Support for Planning Global Software Development Projects," in *IEEE International Conference on Computer and Information Technology* , Xi'an, 2014.

[2] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, Springer-Verlag, 1992.

[3] W3Schools, "Introduction to HTML," [Online]. Available: http://www.w3schools.com/html/html_intro.asp. [Accessed 17 Feb 2016].

[4] w3schools, "CSS Introduction," [Online]. Available: http://www.w3schools.com/css/css_intro.asp. [Accessed 17 Feb 2016].

[5] "JavaScript | MDN," [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/JavaScript. [Accessed 16 Feb 2016].

[6] "Firebase - Build Extraordinary Apps," Google, [Online]. Available: https://www.firebase.com/. [Accessed 5 February 2016].

[7] J. Herbsleb and D. Moitra, "Global software development," *IEEE Softw.*, vol. 18, no. 2, pp. 16-20, 2001.

[8] M. Vanzin, R. Prikladnicki, M. B. Ribeiro, I. Ceccato and D. Antunes, "Global Software Processes Definition in a Distributed Environment," *Annual IEEE/NASA Software Engineering Workshop*, pp. 57 - 65, 7 April 2005.

[9] E. Carmel, Global Software Teams: Colloborating Across Borders and Time Zones, 1 ed., Upper Saddle River, NJ: Prentice Hall, 1999, p. 269.

[10] E. O. Conchúir, H. Holmström, P. J. Ågerfalk and B. Fitzgerald, "Exploring the Assumed Benefits of Global Software Development," in *IEEE International Conference on Global Software Engineering*, 2006.

[11] Y. H. Shah, M. Raza and S. UlHaq, "Communication Issues in GSD," *International Journal of Advanced Science and Technology*, vol. 40, March 2012.

[12] H. Holmstrom, E. O. Conchúir, P. J. Ågerfalk and B. Fitzgerald , "Global Software Development Challenges: A Case Study on Temporal, Geographical and Socio-Cultural Distance," in *International Conference on Global Software Engineering* , 2006.

[13] J. D. Herbsleb and A. Mockus, "An empirical study of speed and communication in globally distributed software development," *IEEE Transactions on Software Engineering*, vol. 29, no. 6, pp. 481 - 494, 20 June 2003.

[14] T. Allen, Managing the flow of technology, Cambridge, MA: MIT Press, 1977.

[15] R. E. Kraut, J. Galegher and C. Galegher, "Patterns of Contact and Communication in Scientific Research Collaboration," in *Intellectual Teamwork: Social and Technological Foundations of Cooperative Work*, Hillsdale, NJ: L. Erlbaum Associates, 1990, pp. 149-171.

[16] A. Mockus and J. Herbsleb , "Challenges of Global Software Development," in *Software Metrics Symposium*, London, 2001.

[17] E. Richardson, V. Casey , M. O'Riordan, B. Meehan and I. Mistrík, "Knowledge Management in the Global Software Engineering Environment," *IEEE*, 2009.

[18] "What is Concurrent Engineering?," UK PTC Reseller and Partner, [Online]. Available: http://www.concurrent-engineering.co.uk/what-is-concurrent-engineering. [Accessed 2 Feb 2016].

[19] I. Richardson, V. Casey, J. Burton and F. McCaffery, Global software engineering: A software process approach, Berlin, Heidelberg: Springer, 2010, pp. 35-56.

[20] J. Noll, S. Beecham and I. Richardson, "GLOBAL INTERCULTURAL COLLABO-RATION: Global software development and collaboration," vol. 1, no. 3, 2010.

[21] C. Wu and D. Simmons, "Software Project Planning Associate (SPPA): a knowledge-based approach for dynamic software project planning and tracking," in *Computer Software and Applications Conference*, Taipei, 2000.

[22] R. Wideman, Project management body of knowledge (PMBOK), Upper Darby, Pa: Project Management Institute, 1987, pp. 39-46.

[23] H. Kerzner, Project management, 10 ed., New York: John Wiley, 2001, p. 412.

[24] C. Jones, "Patterns of Large Software Systems: Failure and Success," *IEEE Computer,* vol. 28, no. 3, pp. 86-87, March 1995.

[25] B. Simmons, "Mathwords: Exponential Growth," Mathwords, [Online]. Available: http://www.mathwords.com/e/exponential_growth.htm. [Accessed 2 Feb 2016].

[26] M. Harman and B. Jones, "Search-based software engineering," *Information and Soft-ware Technology,* vol. 43, no. 14, pp. 833-839, 2001.

[27] M. Harman and A. Mansouri, "Search Based Software Engineering: Introduction to the Special Issue of the IEEE Transactions on Software Engineering," *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING,* vol. 36, no. 6, pp. 737-741, NOVEMBER/DECEMBER 2010.

[28] M. Harman, A. Mansouri and Y. Zhang, "Search Based Software Engineering: A Comprehensive Analysis and Review of Trends, Techniques and Applications," Technical Report TR-09-03, Dept. of Computer Science, King's College London, 2009.

[29] M. B. Cohen and M. O. Cinneide, Search Based Software Engineering: Third International Symposium on Search Based Software Engineering, SSBSE 2011, Szeged, Hungary, September 10-12, Springer, p. 142.

[30] P. Tonella, "Evolutionary Testing of Classes," *Proc. 2004 ACM SIGSOFT Int'l Symp. Software Testing and Analysis*, pp. 119-128, July 2004.

[31] M. Cohen, S. B. Kooi and W. Srisa-an, "Clustering the Heap in Multi-Threaded Applications for Improved Garbage Collection," *Proc. Eighth Ann. Conf. Genetic and Evolutionary Computation,* vol. 2, pp. 1901-1908, July 2006.

[32] A. J. Bagnall, V. J. Rayward-Smith and I. M. Whittley, "The Next Release Problem," *Information and Software Technology*, vol. 43, no. 14, pp. 883-890, 2001.

[33] F. Chicano and E. Alba, "Software Project Management with Gas," *Information Sciences*, vol. 177, no. 11, pp. 2380-2401, June 2007.

[34] M. O'Keeffe and M. O. Cinne´ide, "Search-Based Software Maintenance," *Proc. Conf, Software Maintenance and Reeng.*, pp. 249-260, March 2006.

[35] "Trello," [Online]. Available: https://trello.com/. [Accessed 5 February 2016].

[36] "Google Code Archive," [Online]. Available: https://code.google.com/archive/p/zen-coding/.

[37] w3schools.com, "HTML5 Local Storage," w3schools.com, [Online]. Available: http://www.w3schools.com/html/html5_webstorage.asp. [Accessed 1 4 2016].

[38] developerWorks, "Using the HTML5 IndexedDB API," IBM, [Online]. Available: http://www.ibm.com/developerworks/library/wa-indexeddb/. [Accessed 1 4 2016].

[39] K. Systä, "Seminar TIE-11406: Backend as a Service (BaaS)," [Online]. Available: http://www.cs.tut.fi/~taivalsa/kurssit/BaaS2013/. [Accessed 6 4 2016].

[40] C. Buckler, "HTML5 Browser Storage: the Past, Present and Future," [Online]. Available: http://www.sitepoint.com/html5-browser-storage-past-present-future/. [Accessed 3 4 2016].

[41] "Features - Firebase," Google, [Online]. Available: https://www.firebase.com/features.html. [Accessed 7 Feb 2016].

[42] L. Kwei-Jay and S. H. Son, "Real-time databases: characteristics and issues," in *First Workshop on Object-Oriented Real-Time Dependable Systems, 1994. Proceedings of WORDS 94*, Dana Point, CA, 1994.

[43] "JSON," [Online]. Available: http://www.json.org/. [Accessed 26 FEB 2016].

[44] w3schools, "JSON Tutorial," [Online]. Available: http://www.w3schools.com/json/. [Accessed 26 FEB 2016].

[45] "Installation & Setup - Firebase," Google, [Online]. Available: https://www.firebase.com/docs/web/guide/setup.html. [Accessed 7 Feb 2016].

[46] "Understanding Data - Firebase," Google, [Online]. Available: https://www.firebase.com/docs/web/guide/understanding-data.html. [Accessed 7 Feb 2016].

[47] "JavaScript API - Firebase," Google, [Online]. Available: https://www.firebase.com/docs/web/api/. [Accessed 7 Feb 2016].

[48] "Saving Data - Firebase," Google, [Online]. Available: https://www.firebase.com/docs/web/guide/saving-data.html. [Accessed 7 Feb 2016].

[49] "Retrieving Data - Firebase," Google, [Online]. Available: https://www.firebase.com/docs/web/guide/retrieving-data.html. [Accessed 7 Feb 2016].

[50] S. Vathsavayi, O. S. Korte and K. Systä, Artists, *Functional decomposition of the e-home project.* [Art]. Department of Pervasive Computing - Tampere University of Technology, 2014.

[51] S. Vathsavayi, O. S. Korte and K. Systä, Artists, *Pareto fronts of non-dominated solutions.* [Art]. Department of Pervasive Computing - Tampere University of Technology, 2014.